Annual Technical Report

Covering the Period 1 April 1974 through 31 March 1975

SPEECH UNDERSTANDING RESEARCH

By: DONALD E. WALKER, WILLIAM H. PAXTON, JANE J. ROBINSON, GARY G. HENDRIX, BARBARA G. DEUTSCH, ANN E. ROBINSON

Prepared for:

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY
ARLINGTON, VIRGINIA 22209

CONTRACT DAHC04-75-C-0006
ARPA Order No. 2903
Program Element Code 61101E

STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 · U.S.A.

STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025

Approved for public release;
distribution unlimited.

Form Approved
Budget Bureau No. 22-R0293

June 1975

Annual Technical Report
Covering the Period 1 April 1974 through 31 March 1975
Stanford Research Institute Project 3804

SPEECH UNDERSTANDING RESEARCH

Submitted By

Donald E. Walker
Project Leader
(415) 326-6200, Ext. 3071

With Contributions by Members of the Project Staff

CONTRACT DAHC04-75-C-0006
ARPA Order No. 2903
Program Element Code 61101E

Effective Date: 15 October 1974
Expiration Date: 14 October 1975
Amount of Contract: $299,988

Prepared for

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY
ARLINGTON, VIRGINIA 22209

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

Approved by:

BERTRAM RAPHAEL, Director
Artificial Intelligence Center

BONNAR COX, Executive Director
Information Science and Engineering Division

Copy No. 18

MEMBERS OF THE PROJECT STAFF

Barbara G. Deutsch

Joyce B. Friedman

Gary G. Hendrix

William H. Paxton

Ann E. Robinson

Jane J. Robinson

Donald E. Walker

ABSTRACT

This report is the third in a series of Annual Reports describing the research performed by Stanford Research Institute to provide the technology that will allow speech understanding systems to be designed and implemented for a variety of different task domains and environmental constraints. The current work is being carried out cooperatively with the System Development Corporation, which is responsible for signal processing, acoustics, phonetics, and phonology.

Following an Introduction and Overview, separate sections describe in detail the Definition System, the Parsing System, the Language Definition, Semantics, and Discourse Analysis and Pragmatics. Appendix A contains a listing of the language currently defined in the speech understanding system. Appendix B lists the reports and publications issued by the project staff.

Additional keywords: computer applications

TABLE OF CONTENTS

FIGURES

I INTRODUCTION AND OVERVIEW

Prepared by Donald E. Walker

Contents:

A. Introduction

1. Project Objectives

This report is the third in a series of Annual Reports describing the research performed by Stanford Research Institute (SRI) on the development of a speech understanding system capable of engaging a human operator in a conversation about a specific task domain.[1] This project is part of a five-year program of research sponsored by the Information Processing Techniques Office of the Defense Advanced Research Projects Agency.[2]

------

[1] See Walker (1973a) and Walker (1974a). References are listed in Section VII, at the end of the report.

[2] The rationale for this program and the parameters for the target system can be found in Newell et al. (1973).

The long term objective of the research at SRI on speech understanding is to develop the technology that will allow speech understanding systems to be designed and implemented for a wide variety of different task domains and environmental constraints. Early in 1974, SRI began to work cooperatively with the System Development Corporation (SDC) on the design and implementation of a joint system. The first major step toward the SRI long term objective is completion with SDC of this system in substantial satisfaction of the specifications presented in the Newell Report (1973). We expect to complete by fall 1975 a 'milestone system' that will have most of the components required for the 'five-year' system. This Annual Report describes the contributions that have been made by SRI to the development of this milestone system.

2. Background

For three and a half years, SRI has been participating with other ARPA/IPTO contractors in a major program of research on the analysis of continuous speech by computer. During the first year of the SRI project, the domain chosen provided interactions with a simulated robot that knew about and could manipulate various kinds of blocks.[3] The system implemented during this period made major use of procedures developed by Winograd (1971) for understanding sentences in natural language entered as text.

------

[3] For descriptions of these initial efforts, see the First Annual Report for the project (Walker, 1973a), Walker (1973b), and Paxton and Robinson (1973).

During the second year of the project, a new task domain was chosen: the assembly and repair of small appliances. This change was made to provide for more complex interactions of a user with the system, entailing a sequence of goal-directed subtasks. Major modifications were made in all parts of the system, the most important of which was the development of a new parsing strategy.[4]

SRI began collaborating with SDC on the development of a system following the Midterm Evaluation of the total ARPA Speech Understanding Research Program. Because of the similarity of the design concepts for the two contractors, it has been possible to combine features and components of the two most recent systems of each in building the new system architecture.[5] Work on signal processing, acoustics, phonetics, and phonology at SDC is being coordinated with work on parsing, syntax, semantics, pragmatics, and discourse analysis at SRI. There is shared responsibility for system design, for the specification of task domains, and for work on prosodics.

Two task domains have been selected for the duration of the current five-year program:

------

[4] See the Second Annual Report for the project (Walker, 1974a); also see Walker (1974b), Paxton (1974), Becker and Poza (1975), Deutsch (1974), and Robinson (1975). Walker (1973c) provides a perspective on the transition from the first to the second versions of the system.

[5] For an overview of the previous SDC efforts and references to other SDC papers, see Ritea (1974).

(1) Data management of a file containing information about selected ships from the fleets of the United States, the Soviet Union, and the United Kingdom.

(2) Maintenance of electromechanical equipment in a workstation environment with the system as a computer consultant.

Since we began working with SDC, most of our activities have concentrated on the first domain, but a substantial amount of effort has gone into ensuring the generality of our system structure and its appropriateness to the second domain.

The task domains selected are significantly different in kind. Together, they represent the two major kinds of knowledge identified in artificial intelligence research: state knowledge and process knowledge. State knowledge captures information about a static world, all the facts that hold at a particular instant in time or for all time. Retrieving information from a formatted file is a representative task over state knowledge. Process knowledge embodies a dynamic model of the interrelations among the elements of a world so that change over time can be handled directly. Repairing an air compressor or a jeep exemplifies a relevant task.

The work on the second task domain is complemented by the activities of a companion project at SRI that is developing a comprehensive 'computer based consultant' (CBC) system.[6] That system is designed to guide a technician in the maintenance of electromechanical equipment in a workstation environment. Our speech understanding system can provide the basis for communication with the computer in natural language.

B. Overview of the System

1. Introduction

An initial version of the cooperatively developed speech understanding system has been implemented and tested at SDC. The acoustic processing is provided by the Raytheon 704, and the rest of the system is programmed in SDC/LISP on the IBM 370/145. In addition, the parser and the syntactic, semantic, and discourse components have been exercised extensively at SRI on the PDP-10, with simulations of the acoustic, phonetic, and phonological components. These versions of the higher level language components are programmed in INTERLISP. More extensive testing of the total system will be conducted when INTERLISP/370 is available on the IBM 370/145 and when other components are reprogrammed for that computer in CRISP, a new programming system now under development at SDC. For the milestone system, SDC will replace the Raytheon 704 with an acoustic preprocessor consisting of a PDP-11/40 and an SPS-41 special purpose digital signal processor.

------
[6] ARPA Contract No. DAHC04-75-C-0005, SRI Project 3805. See Nilsson et al. (1975) for the most recent Annual Report and Hart (1975) for an overview of the project.

The following summary provides a perspective on the distinctive characteristics of the ~~SRI~~ Stanford Research Institute contributions to the current system. The system control, embedded in the parser, focuses the operation of the entire system to minimize both storage requirements and the time spent on incorrect interpretations. A language definition system provides a means for integrating the various sources of knowledge in the system. The language definition itself, based on studies of protocols gathered from actual performances in task-oriented dialogs, includes information from acoustics, phonetics, phonology, prosodics, syntax, semantics, pragmatics, and discourse. A new semantic network representation, which partitions the net into spaces, has proved particularly well suited for working with the two task domains. Discourse procedures, building on the semantics, establish a discourse history so that information from previous utterances (and, ultimately, from the task environment) can be used in the analysis of the current utterance. Descriptions of these developments and of the work in progress are presented in the rest of this section. These presentations will serve as an introduction to the sections on the various system components that constitute the major part of this Annual Report.

2. Status of the System Components

a. System Control

The parsing system coordinates and controls the other system components in the process of understanding an utterance. A computationally efficient internal representation of the various knowledge sources is established through the language definition system, providing a uniform way of integrating different kinds of information. The external representation of the language definition is described under item 3 below. Words and phrases can be predicted on the basis of context, and phrases can be built up from words that have been identified acoustically in the utterance.

During the search for a complete interpretation of the utterance, a complex data structure called a 'parse net' is built up. The various tasks corresponding to alternative analyses are assigned priorities and scheduled according both to their estimated value and to a focus of activity that takes into consideration processing time and current storage requirements. When the performance of a task results in the prediction of a word at a specified place in the utterance being processed, various alternative phonological forms of that word are mapped onto the acoustic data for that place, and a score denoting the degree of correspondence is returned. Subsequently, when a phrase containing that word is predicted, another mapping is done to take into account coarticulation effects of the words on each other.

The parser stops and calls a response function when it has an interpretation for the entire utterance or when it reaches a specifiable limit either on the number of tasks to be performed, on the lowest value of a priority it will accept, or on the amount of space it can use.

Efficiency has been a major motivating factor in the design of the parsing and language definition systems, with respect both to the effort required by the people who are entering data and to the actual computations carried out inside the computer. A language definition compiler automatically converts rules as a linguist would write them into a form optimal for machine processing. The parse net brings together work on common substructures to eliminate duplication of effort. In addition, the various ways in which the same information can be used in different internal operations are anticipated, and, for computational efficiency, separate representations are constructed that are optimal for each use.

The two elements of system control, the language definition system and the parsing system, are presented in detail in the sections with those headings.

b. Language Definition

The subset of natural spoken English that the system is designed to understand is specified by the language definition (LD). This component in a question-answering system is usually called a 'grammar', but our LD takes into account such a variety of linguistic information that 'grammar' does not adequately encompass it. The LD has two major parts:

(1) A collection of basic units, called 'word definitions' (WD), which correspond roughly to words and together form a lexicon.

(2) A collection of definitions of rules, called 'composition rule definitions' (CRD), for combining words and phrases into larger units.

Each CRD contains statements that assign attributes to the resulting unit based on available acoustic, phonetic, phonological, prosodic, syntactic, semantic, pragmatic, or discourse information. A CRD also contains factor statements that establish how well the resulting unit fits the corresponding part of the utterance, on the basis of all the determinable attributes.

Since October 1974, the language definition has been extended, as well as refined, to adapt it to the discourse found in protocols collected for the data management task domain during the summer and fall. (Before that time, it defined a language we assumed would be relevant for querying a small data base drawn from Jane's Fighting Ships.) New definitions were added for elliptical utterances and for limited comparative expressions involving numbers. Pragmatic factors were added to existing definitions to adapt the LD to the high frequency of WH-interrogatives and elliptical nominals. By the end of 1974 more than 60 phrase types and 30 syntactic categories had been defined, and the LD had been tested extensively on text and simulated acoustic input and in a limited fashion on actual acoustic input.

Further extensions to the language definition are being made on the basis of analyses of additional protocols from both task domains. CRDs are being written for additional phrase types that are typical of the discourse required for the tasks and sufficiently tractable to be put into the system and tested in a reasonable time. These include definitions for some kinds of quantification, limited coordination, relative clauses, and compound nominals. They will be ready for testing by the end of the current contract.

Earlier this year, SRI and SDC, together with the Speech Communication Research Laboratory (SCRL), established a set of conventions for transcribing protocols from our task domains, marking pauses (both silent and 'filled'), tonic syllables, and pitch direction. The data from these transcriptions are being used to revise the prosodic statements currently in the LD.

Further work on prosodics will be based on our judgment that the acoustic phenomena promising the most immediate returns for a prosodic component are silence and duration. The matrix of acoustic and phonetic data for one of the early submarine protocols was handmarked to locate pauses and to identify word durations. A concordance was compiled that brought together, in context, all occurrences of each word and pause, in order of increasing duration. These data allowed us to make comparisons and form hypotheses regarding the distribution of pauses and, in particular, the correlation of pauses with word boundaries and with word durations. We are arranging to test these hypotheses on the next round of protocols from different speakers. Our first comparisons support observations reported in the general literature on prosodics, which indicate that it should be possible to specify minimal durations for some kinds of words (stressed 'content' words) and conditions on lengthening of unstressed words before pause.

We plan to use other acoustic attributes of words to distinguish among a set of words that are predicted for a particular place in the utterance. A preliminary scheme for classifying words on the basis of strong acoustic clues in their initial and final syllables has been developed. It is now being implemented at SDC and will be tested during the summer. Simulated tests on text with and without this lexical subsetting capability lead us to expect a significant improvement in parsing efficiency. Adding prosodic cues to the procedure should increase its discriminatory power.

A description of the current state of the language definition and examples of the utterances it can handle are presented in Section IV, The Language Definition. Appendix A contains a complete listing of the language definition in the format prescribed for actual use in the speech understanding system.

c. Semantic Analysis

The semantic component that has been developed for our speech understanding system consists of two major parts:

(1) A semantic network coding a model of the task domain.

(2) A battery of semantic composition routines that are directly coordinated with the language definition to build network representations of utterances.

Our semantic nets differ from other network representations in that the nodes and arcs of our nets are partitioned into units called spaces. These spaces group information into bundles that help to condense and organize the semantic knowledge base of the system. Specifically, partitioning facilitates quantification, which in turn makes possible the description of generalized categories of objects, situations, and events. The organization of knowledge in terms of hierarchies of categories results in a more economical storage of information with properties common to all elements of each category being stored only once at the category level. (It remains clear that these properties are properties of the category members and not of the category itself.)

Net partitioning also provides a uniform mechanism for distinguishing hypothetical and imaginary situations from reality, a property of considerable importance in dealing with dynamic domains (such as our computer consultant task) characterized by multiplicities of alternative future states. The semantic composition routines that form a part of each language definition rule call on the information in the network to help understand the meaning of each phrase. Outputs from these routines are network fragments whose structures follow the same encoding conventions followed in the encoding of knowledge in the rest of the network.

We are currently experimenting with an improved set of network manipulation functions that are more efficient than their predecessors and that allow the network to be divided up in multiple ways. One of the new network groupings is being used to establish contexts (and hierarchies of contexts) within the net for use in discourse analysis. The revised network functions also are being integrated into the semantic composition routines in a way that will eliminate both the need for the 'intermediate language' used in our previous work and the need to copy portions of the network in cases of ambiguity or uncertainty.

While modifying the semantic composition routines to use the revised network manipulation functions, several other improvements are being made as well. Our present system uses sequences of code especially written for each verb to associate surface cases with deep semantic cases. These code sequences are being replaced by a two-way case mapper that will interpret a brief statement of case information included with the entry of each verb-like member of the exicon to map from surface into deep cases and vice versa. The added ability to map from deep to surface cases will facilitate semantic prediction and the generation of answers to questions. Other additions to the composition routines currently being developed will provide the following capabilities:

(1) Construction of network representations of phrases for which some constituents are partially or totally unspecified.

(2) Prediction of the composition of the missing components in these incomplete phrases.

One of the most important of our current activities in semantics is designing and implementing a retrieval system that will examine the network structure produced as a result of parsing, interpret the meaning of the input, and develop and execute a plan for producing an appropriate response. Our short term goal is tc respond appropriately to input queries that contain only one verb-like structure and that can be answered from information contained explicitly in the data base. Outputs initially will be YES, NO, or simple noun phrases.

Further details of the work on semantics are provided in Section V, Semantics.

d. Discourse Analysis and Pragmatics

During the current contract period, we continued to collect and analyze protocols of task-oriented dialogs. Previously, with the cooperation of personnel from the Naval Postgraduate School in Monterey, California, we had conducted experiments for the data management task domain in which naval officers queried specifications and performance characteristics of submarines in the U.S., Soviet, and British fleets. Also, in conjunction with the computer based consultant project at SRI, we gathered dialogs from the workstation environment for our second task domain. Currently, with the help of the Naval Electronics Laboratory Center in San Diego, we are recording protocols using a new data management scenario involving U.S. and Soviet ships in the Mediterranean. Further experiments for the computer consultant task are planned.

The protocols already gathered have been analyzed to identify modifications in the syntax and vocabulary, as described in the section on language definition. In addition, they have been examined for instances of ellipsis and anaphoric reference. Guided by this analysis, we have designed and implemented a preliminary discourse package that handles the simpler forms of ellipsis and anaphora found in the dialogs. A history of previous utterances is kept, and after an utterance is successfully parsed, references are resolved using the immediately preceding utterance as context. In addition, elliptical utterances are completed, if possible, by comparing them with parts of the preceding utterance and adapting the structure in which the corresponding parts are embedded.

We are in the process of augmenting these cedures in several ways. The availability of multiple partitions and contexts in the semantic net will enable us to identify 'focus spaces', that is, regions that are directly related to the current discourse. Use of this mechanism will limit the portion of the net that has to be considered in resolving references; it will be particularly helpful for the computer consultant task domain, which is more structured and considerably more complicated than the data management one. Four steps are entailed in making these extensions:

(1) Representation of the focus partition in the semantic network.

(2) Preparation of a set of criteria for deciding when to establish a new focus space and what to put in it.

(3) Development of a set of heuristics for deciding which spaces to search and when to search them in order to resolve uncertainties of reference.

(4) Integration of the focus space mechanism with the current discourse package.

In the milestone system, we expect to be resolving simple anaphoric references from the discourse context using the focus space structure. Furthermore, resolution will be performed at the phrase level rather than waiting until the structure for a complete utterance has been produced. We also will introduce a preliminary form of prediction on the basis of the discourse routines. The discourse history will be extended to keep track of topics recently talked about, and procedures will be developed to change the priority (score) of related elements in the language definition accordingly.

A detailed examination of these activities is provided in Section VI, Discourse Analysis and Pragmatics.

II THE DEFINITION SYSTEM

Prepared by William H. Paxton

Contents:

A. Introduction

Research on natural language understanding faces the problem of developing a definition of the structure and content of a language such as English in a form allowing efficient use by a computer. Systems for representing such definitions are judged according to two sets of criteria. First are the 'human-motivated' criteria of simplicity, generality, modularity, and the like. These are obviously important requirements in a representation system that must support the development of a large, complex definition. The criteria in the second set are 'computer-motivated' and relate to efficiency in use of time and space resources. These requirements come from the fact that the definition must eventually be put to use in an operational understanding system.

Unfortunately, the two sets of criteria tend to conflict. For example, clarity of the definition for the person writing and rewriting it is an important criterion from the first set. A major step toward clarity is to devise representations structured such that redundant information is factored out and stated once rather than repeated throughout the definition. However, this approach to clarity tends to conflict with the desire for efficient operation since efficiency can often be enhanced by redundant representations that anticipate common modes of access and use.

Such conflicts suggest that any attempt to satisfy the two sets of criteria in a single representation system must ultimately be unsatisfactory. A well-known alternative is to have two representation systems: one for an "external" form of the definition for use by people and primarily reflecting the first set of criteria, and another fo. an "internal" form of the definition for use by the computer and emphasizing the second set of criteria. The representation systems must be compatible in the sense that the internal definition must be (automatically) derivable from the external definition, but otherwise they are independent.

In line with this alternative, we have developed a definition system composed of a Language Definition Language for writing the external representation and a Language Definition Compiler to translate to the internal representation. These two components are discussed in detail in the remainder of this section. The actual use of the internal representation in parsing is covered in Section III, The Parsing System.

Before describing the language definition language in detail, the effects related to the required compatibility with the internal representation are discussed. This discussion also serves to introduce one of the most distinctive features of the system--the prominent role of factors relevant to the choice of an interpretation for an input utterance.

As mentioned above, it must be possible to translate the external form of the definition into an efficient internal representation for use in parsing. A primary effect of this requirement is to rule out a number of possibilities for the external representation system. Among those excluded for lack of an efficient implementation method are some that have been popular in contemporary linguistics concerning transformations of phrase markers. (This characterization applies equally to Chomsky's various formulations and to rival approaches such as generative semantics; see, for example, Chomsky, 1971, and Lakoff, 1971). While their transformational rules are not used directly, the data and insights of generative grammarians are obviously valuable to anyone with the goal of constructing a system to understand natural language.

A second effect of the compatibility requirement relates to the content of the definition rather than its form. Information required as part of the internal representation of the language must be either explicitly contained in the external definition of the language or deducible from it by the compiler. This implies a feedback from decisions about the parsing system and its operation to decisions about the content of the language definition. Such feedback suggests that at least part of what you know when you know a language is information that makes possible efficient processing of utterances in the language. What sort of information is this?

In trying to understand an utterance, the parsing system is continuously faced with choices--what word was said here? what kind of construction was used there? what does this phrase refer to? what does that one mean? If the processing is to be efficient, the choices must be made wisely. To make wise choices the parser needs access to information about the language that goes beyond the traditional distinction between grammatical and ungrammatical. Before it can choose among competing alternatives, the parser must determine their relative 'values'.[1] In general, many factors must be considered: contextual factors based on the linguistic and nonlinguistic environment of the utterance, structural factors based on syntactic, semantic, and stylistic interrelations, and acoustic factors based on the actual input signal and such things

------
[1] The 'value' of an alternative is used here as a technical term simply meaning a measure used as a basis for choice. As such, it should not be assumed to correspond to some other formal measure such as probability.

as the phonological and acoustic-phonetic rules of the language. Thus instead of always reflecting categorical, yes-or-no restrictions, factors may have a wide range of possible values based on probabilistic tendencies as in the case of stylistic variation, or on uncertain information as in the case of acoustic segmentation and classification.

Discussions are given below of how such factors are represented in a language definition and how their values are merged to produce a composite evaluation. The use of the factors to guide the processing of an utterance is a major topic of Section III, The Parsing System.

B. Language Definition Language--External Representation

A language definition includes sets of units out of which utterances in the language are constructed, rules for combining the units into larger structures, and general statements about the units, rules, and other aspects of the language. The basic units will be called 'words' (although this technical use does not exactly correspond to the common use), and the total set of words will be referred to as the 'lexicon'. The lexicon is partitioned into categories such as noun and verb, and the words in each category are assigned values for various attributes such as phonological form, grammatical features, and semantic representation. For every lexical category there is also a definition specifying attributes and factors that must be computed for each particular occurrence of a word of that category in an utterance.

A second major part of a language definition is the set of composition rules that indicate how words can be combined into phrases. More precisely, a 'phrase' is either a word in the input or the result of applying a composition rule to constituent phrases. The rules give the linear pattern of constituents and specifications for calculating values of both the attributes of the resulting phrase and the factors to be considered in judging the result. Finally, a complete language definition also includes a set of global declarations such as lists of categories and their attributes and redundancy rules to be applied in translating other parts of the definition. The redundancy rules state general properties of the language so that the properties do not have to be (redundantly) repeated throughout the definition.

1. Composition Rules

Figure II-1 contains a composition rule that might occur as part of a definition for a subset of English. (The rule is actually a simplified version of a rule occurring in the language definition; see Appendix A.) The rule defines 'Yes-No' questions like "Is that thing a rule definition?" made up of a form of the verb 'be' and two noun phrases. The first line of the rule starts with the keyword RULE.DEF indicating that a rule definition follows. The rest of the first line gives the name of the rule, S8, and the composition pattern for the rule. The pattern indicates that phrases built according to this rule will be of category S and will be composed of three constituents, the first of category AUXB, the second and third of category NP. The remaining parts of the rule need names with which to refer to the constituents. Often the category name can serve as the name of the constituent; for example, AUXB will be the name of the first constituent in this rule, but when the constituent category is not unique, a name must be given explicitly. Thus in this rule the first NP is given the name NP1 and the second, NP2.

Figure II-1 A Simplified Composition Rule

```
RULE,DEF S8     S = AUXB NP:NP1 NP:NP2;
  ATTRIBUTES
    RELN,CMU,FOCUS FROM NP1,
    MOOD = "(YN),
    SEMANTICS = SEMCALL("SEMRS8,SEMANTICS(NP1),SEMANTICS(NP2)),
    PITCHC = FINDPITCHC(PLEFT,PRIGHT);
  FACTORS
    GCASE1 = IF GCASE(NP1) EQUAL "(ACC) THEN OUT ELSE OK,
    GCASE2 = IF GCASE(NP2) EQUAL "(ACC) THEN OUT ELSE OK,
    MOOD1 = IF MOOD(NP1) EQUAL "(WH) THEN BAD ELSE OK,
    MOOD2 = IF MOOD(NP2) EQUAL "(WH) THEN BAD ELSE OK,
    NBRAGR1 = IF CMU EQUAL "(UNIT) THEN
      [IF NBR(AUXB) EQUAL "(SG) THEN OK ELSE OUT]
      ELSE IF GINTERSECT(NBR(NP1),NBR(NP2)) THEN OK ELSE OUT,
    NBRAGR2 = IF CMU(NP2) EQUAL "(UNIT) THEN OK ELSE
      IF GINTERSECT(NBR(NP2),NBR(AUXB)) THEN OK ELSE OUT,
    PERSAGR = IF GINTERSECT(PERS(NP1),PERS(AUXB))
      THEN OK ELSE OUT,
    FOCUS = IF FOCUS(NP1) EQ "INDEF AND FOCUS(NP2) EQ "DEF
      THEN POOR ELSE OK,
    RELN = IF RELN EQ "T THEN
      IF CMU EQUAL "(UNIT) THEN VERYGOOD ELSE OK,
    SCORE IF NOT VIRTUAL,
    STRESS = IF VIRTUAL THEN OK ELSE
      IF STRESS(AUXB) EQ "UNREDUCED THEN GOOD,
    PITCHC = IF VIRTUAL THEN OK ELSE
      IF PITCHC EQ "HIRISE THEN GOOD ELSE OK;
  END;
```

Following the pattern is a set of statements specifying attributes of phrases constructed according to this rule. Some attributes always have the same value; for example, in this rule the MOOD attribute is always (YN), meaning a Yes-No question. Other attributes are simply the same as the corresponding attribute of one of the constituents; in this rule, FOCUS is taken from the FOCUS attribute of the constituent NP1. In general, however, attributes are calculated on the basis of attributes of constituents, as in the case of SEMANTICS, which depends in a complex way on the SEMANTICS attributes of the constituents. In addition to the ones explicitly given in the rule definition, other attribute statements are supplied by redundancy rules. Among others, the LEFT boundary attribute (always derived from the leftmost constituent) and the RIGHT boundary attribute (always from the rightmost constituent) are added in this manner.

Following the attribute statements is another set of statements specifying some of the factors to be considered in evaluating phrases built by this rule. The factors include syntactic considerations such as case, mood, number, and person agreement. Other factors raise the evaluation if the auxiliary verb is phonetically unreduced or if the pitch rises at the end of the sentence. Redundancy rules add still more factors such as one to check coarticulation effects among the constituents and another to reduce the score if no semantic representation can be found.

Attributes and factors either have constant values or depend only on attributes of constituents and global information such as a model of the discourse or the results of preliminary, low-level acoustic processing. By design, the attributes and factors for a phrase are not allowed to depend on the context formed by other phrases actually or potentially combining with it to form a larger structure. Context-sensitivity of this type is not permitted since it tends to introduce assumptions about the parsing strategy into the language definition. An example will help to illustrate this. A noun phrase can be composed of an article followed by a nominal phrase, in which case the article and the noun must agree in various ways such as plurality. In a system allowing restrictions to refer to contextual features, article and noun agreement might be ensured by having nouns check to see that they are preceded by an article with appropriate features. The potential problem with this procedure is that the test is easy to implement if the article is always available by the time the noun is reached and the restriction is to be checked, but this depends on details of the parsing strategy. The parser must either ensure that the relevant contextual information is unambiguously available by the time the test is to be made or take on the burden of remembering to perform the test at some later point when the necessary information does become available.

Both the above options are unattractive; the first, because it limits the possibilities for the parsing strategy and forms strong ties between the language definition and the particulars of the parser, ties that make change difficult; the second, because it promises to add substantial complexity and overhead to an already complex and costly parsing process. An alternative that avoids these objections is to put the restriction with the rule that brings the article and the noun together rather than with either the noun or the article individually. There are then no assumptions about whether the article is found first and used to constrain the choice of a noun, or the noun first constraining the article, or both independently with a separate test to eliminate bad combinations. The language definition simply states the restriction and is neutral with respect to its use in parsing. It was to foster this neutrality that attributes and factors were made to refer only to global data and attributes of constituents rather than to sentential context.

Another significant property of rule attributes and factors is that their definitions must cover cases in which the value of a referenced attribute, in a sense, is undefined. Specifically, if rule factor F is calculated using attribute A (from the same rule definition or from a constituent), then the algorithm for computing F must produce a reasonable result even if A has the special value 'UNDEFINED'. Similarly, if rule attribute B uses attribute A in its definition, then the algorithm for B must give a reasonable result if A is UNDEFINED. For a factor, a reasonable result would be either an estimate of its best value if A had been defined or a special 'don't care' value keeping it from influencing the overall score (the scoring function for combining factors is discussed below). For an attribute, a reasonable result would be either a value indicating the range of possible outcomes if A had been defined or simply UNDEFINED to propagate the lack of information.

There are several reasons for requiring attribute and factor statements to deal with UNDEFINED attributes. First, the various rules and words that can be used to form a particular type of constituent may differ in the attributes they define. For instance, attribute A may be defined in rule 1 but not in rule 2. Rather than insisting on an explicit definition for A, the system instead causes A to be UNDEFINED in any phrase constructed by rule 2. Other rules that reference attribute A of a rule 2 phrase will find it to be UNDEFINED, accurately reflecting the fact that rule 2 did not include a definition for A.

Another motive for UNDEFINED attributes is the aim of extending the system eventually to deal with utterances containing words not included in the lexicon. If the structure of the utterance, as constructed by the parser, suggests that the unknown word is a noun, say, then the word can be tentatively entered in the lexicon as a noun with all attributes UNDEFINED. Since the rules of the language definition allow such attributes, the new word can be used as part of larger phrases. Moreover, if the system successfully produces a complete parse using the new word, then it should be possible to make provisional assignments to various UNDEFINED attributes by looking for values that would

Yr

yield the best results for dependent factors in phrases containing the word. For example, on the basis of sentence 1, anyone who knows English can guess that "frammus" is a noun referring to a small physical object that is probably edible.

(1) The little dog ate the frammus in a single bite.

As a mechanism for dealing with unknown words, this is still speculative, but it appears to offer an interesting approach worth further study.[2]

The final reason for UNDEFINED attributes is the desire to allow parsing strategies that depend on information regarding incomplete phrases--phrases missing one or more constituents. With the restrictions on attribute and factor statements outlined above, references to attributes of missing constituents can be given the value UNDEFINED, and the results will be indicative of possible completions of the phrase. The use of this ability in the current parser is a major topic in the discussion of the parsing system.

2. Lexicon

Figure II-2 shows a sample lexical entry, that is, a word definition. The definition for "it" is in the set of word definitions for category NP. The attributes given for "it" include syntactic features such as number (singular), person

------

[2] A related mechanism for "learning" vocabulary is sketched in Thorne et al. (1968).

(third), and mood (declarative as opposed to interrogative). The entry also has information to be used in producing a semantic interpretation (the WDSEMANTICS attribute). The attribute values of a lexical entry, whether given explicitly in the entry or derived by redundancy rules, are shared by all instances of that word.

Figure II-2 The Lexical Entry for "It"

```
IT
  MOOD = (DEC),
  FOCUS = (DEF INDEF),
  GCASE = (NOM ACCUSE),
  CMU = (COUNT MASS UNIT),
  SUBCAT = PRO,
  NBR = (SG),
  PERS = 3,
  WDSEMANTICS = (AMBIGUOUS ((SUPSET  UNIOBJS)(NBR S) (ISF  ISF))
   ((SUPSET  UNIOBJS,MASS)(NBR M)(ISF ISF)));
```

Attributes that vary from one instance to another are specified in a "category definition" that is similar to a rule definition. For instance, the category definition for NP states that the SEMANTICS attribute is to be computed from the WDSEMANTICS of the particular word. This makes it possible to construct different objects, nodes in a semantic network in this case, for different instances of the word. Redundancy rules add other token-dependent attributes such as the positions of the left and right boundaries of the word in the utterance. The category definition also includes a set of factor statements to be used in evaluating potential instances of the category. The most important factor is the match between the input signal and the expected form of the word, and, in the case of speech understanding, determining a value for this factor can be an enormously complex operation. Other factors may eventually be added to the language definition, reflecting such things as expectations regarding how well the word fits into the current topic of conversation or its use by the current speaker. The form of category definitions, their internal representation, and their use in parsing are discussed further below.

The previous example entailed the definition of the word "it" taken from the set of lexical entries for the category NP (noun phrase). Noun phrases can also be constructed by composition rules, such as a rule allowing a determiner and a noun to come together to form a phrase like "this phrase". Both the NP category definition and the NP composition rules produce structures that can potentially be used in contexts calling for a noun phrase. The ability to have both lexical entries and composition rules for a single category simplifies the language definition by removing the need for superfluous categories or patterns such as NP=IT, and allows us to represent certain elementary operations such as the formation of plural nouns.

In English, most nouns are marked for plural by a suffix whose realization depends on the phonological ending of the noun, but which has little effect on the pronunciation of the noun itself (e.g., part:parts, word:words, language:languages). The

SRI language definition captures this regularity by constructing both plural and singular nouns from a separate category, N, of noun stems. Plural nouns come from noun stems by adding the plural suffix, while singular nouns come from noun stems without changing phonologically. Irregular plural nouns are entered as nouns in the lexicon and have their noun stems marked to block the regular pluralization rule. Independent justification for the noun stem category N comes from its appearance in rules for prenominal modifiers. Thus in a phrase such as "this four word phrase", "word" is a noun stem that is neither singular nor plural, rather than a singular noun somehow managing to coexist with the plural modifier "four". That "four" does not modify singular nouns can be seen in an ungrammatical sentence like "Say four word."

This approach to representing simple morphological processes can also be used with other categories such as verbs (suffixes for tense, number, progressive, and passive) and numbers (suffixes "-teen" and "-ty"). Ordinals such as "eighteenth" and "eightieth" illustrate the possibility of adding multiple suffixes, and possessive constructions as in "the man on the street's opinion" demonstrate the need to add suffixes to entire phrases as well as to single words. In general, composition rule patterns can optionally include an affix at the beginning (in which case it is called a prefix) or at the end (when it is called a suffix). The affixes are distinguished from the other parts of the composition pattern in that they are not independent constituents. This influences their treatment in the language definition (they do not occur in the lexicon and do not have attributes or factors apart from the larger structures in which they occur) and in the parsing process (there is no attempt to recognize them apart from the constituent(s) to which they are attached).

3. Global Declarations

In addition to composition rules and a lexicon, a language definition includes a set of global declarations such as the one in Figure II-3. These declarations appear at the beginning of a language definition and are used in the conversion to the internal representation. There are lists of the categories, affixes, and attributes that will be used in the definition, names of redundancy rules for words, lexical categories, and composition rules, the name of the root category of the language (the category for representations of entire utterances), and the name of a response function to be called when the parser constructs instances of the root category.

Currently, redundancy rules are not defined within the language definition system itself, but are instead simply LISP functions that operate on list structures forming an intermediate representation of the definition. (Because of this implementation, they are subsequently referred to as 'redundancy functions'.) For example, the redundancy function for composition rules is called with a list structure representing a rule definition and returns a possibly modified list structure that the compiler then converts into internal form. This way of capturing generalizations is certainly better than nothing, but should eventually be replaced by an extension of the language definition facility so that redundancies can be stated in a language similar to that used in composition rule and lexical category definitions. This will allow the statement of redundancy rules without the distracting details of list processing and representations used during the conversion to an internal form.

Figure II-3 Global Declarations

```
LANGUAGE.DEF
  CATEGORIES U, N, NOUN, NP, DET, VERB, AUXB, VP, S, TOKEN;
  ROOT CATEGORY U;
  AFFIXES PL, TY, TH, GEN;
  RULEFN R;
  WORDFN W;
  CATEGORYFN C;
  RESPONSEFN RS;
  ATTRIBUTES
    ALL HAVE LEFT, RIGHT, SPELLING;
    ALL EXCEPT TOKEN HAVE SEMANTICS;
    S, VP HAVE VOICE;
    U, NP HAVE ELLIPSE;
    S HAS PITCHC;
    ENDATTRS;
  END;
```

In addition to extending the language definition language to allow replacing the current redundancy functions, it will also be important to study ways of including other types of general statements. The definition facility now includes redundancy functions for composition rules, lexical category definitions, and word definitions. The current redundancy functions make changes throughout the entire language definition, but the redundancies reflected by the changes are all local in the sense that they are limited to modifications within items, such as rules or category definitions, that already exist in the external form of the definition.

The modifications are usually additions of factors and attributes and depend only on the properties of the single item under consideration. A good example is the rule redundancy function that looks at the composition pattern of the rule and on that basis alone adds the left and right position attributes. These changes depend on properties local to a single rule and influence only properties local to that rule. There are no redundancy functions that change a group of definition items in a manner that depends on properties of the group as a whole. For instance, no redundancies in the current system depend on properties of the set of all composition rules or modify that set by adding or deleting rules. This lack undoubtedly reflects an area where the definition system needs to be extended rather than an absence of global redundancies worth stating. For instance, perhaps one or more global redundancy rules (GRRs) could be used to state the structure of sentences containing existential "there" in terms of modifications to a language definition not including such constructions.

It would be interesting to compare such GRRs, which act to transform the set of rules defining the language to the transformational rules of generative grammar, which act to transform phrase markers during a derivation. Unlike the latter type of transformation, GRRs would be applied when the language was 'internalized' by the system and would not qualitatively complicate the activity of the parser. Through an attempt to specify GRRs we might come to understand more clearly how complex systems of interrelated rules can evolve as must happen when an individual acquires a language or during other complex learning tasks.

The desire to simplify the development of more powerful redundancy rules is a principal reason behind the choice of a very simple form for constituent pattern specifications in composition rules. The patterns are restricted to series of one or more constituents with the optional addition of affixes. There are a variety of ways in which this form might have been extended. Some of the possible extensions are to allow constituents to be marked as optional, to specify a list of alternatives for a particular portion of the pattern, and to provide an iteration operator indicating zero or more occurrences of a certain constituent. All of these can simplify the statement of patterns, but at the expense of other aspects of the definition. The factor and attribute statements would have to account for each of the cases merged together in a more complex pattern, and redundancy rules would also have to become more complicated. While this is certainly a possible area for change, experience to date supports the correctness of the decision to restrict the form of patterns so as to simplify other parts of the definition.

4. Combining Factors into Composite Scores

Categorical (yes or no) factors act as restrictions on the language--some phrases are disallowed while all others are accepted. There are no in-between cases, no fuzzy areas. Factors of this type can be combined in a simple manner; either they are all satisfied or the phrase is rejected. However, not all factors relevant to evaluating a phrase are categorical; there are many that have a wide range of possible values either because they rely on uncertain information or because they reflect tendencies that are statistically significant but not absolute. Unlike absolute restrictions, such multivalued factors cannot be combined by simple conjunction. To facilitate experimentation with different techniques for combining factor values into a single, composite evaluation (or 'score'), the system has been purposely designed to minimize the assumptions made about the scoring method.

The main assumption has to do with the range and structure of scores. A score must be either an integer between 0 and 100, or a pair of integers of the form <WEIGHT, TOTAL> such that TOTAL divided by WEIGHT is in the range 0 to 100. (The motivation for the latter form of scores is given below.) Upper and lower bounds on scores are needed so that the system can differentiate a good score from a bad one. The only other assumption is that if any factor has a zero value, the resulting score will be so low that the phrase can be discarded. The system checks for this special value; if a factor produces a zero value, the evaluation process stops without unnecessarily calculating the remaining factors.[3]

There are no other assumptions about factor values or their influence on the resulting score. Factors do not even have to evaluate to numbers; they can provide whatever type of information is chosen for use by the score function. For instance, rule factors can refer to the scores of constituents so that the score for a phrase will reflect the constituents individually as well as their interdependencies. Finally, to allow still another area for experimentation on scoring, the system provides for different algorithms for calculating scores to be used in different parts of the language. Each composition rule and lexical category can have its own score function for combining factors.

Within the loose bounds set by the assumptions built into the definition and parsing systems, a score function has been

------

[3] Since the factors are evaluated in the order that they appear in the definition, given estimates of each factor's likelihood of yielding a zero value and cost of evaluation, the factor statements in the definition can be ordered (by the rule redundancy function, say) to minimize the expected total cost by sorting them according to increasing quotient of cost over likelihood of zero value (likelihoods in the range 0 to 1). This means that if two factors have the same cost, the one more likely to produce a zero goes first; if two factors have the same likelihood of producing a zero, the less costly one goes first, and all factors with no chance of producing a zero follow those that can.

developed with the following properties:

(1) If all factors are high, the score is high.

(2) If any factor is very low, the score is low.

(3) As long as no factor is 0, a change in any factor will cause a corresponding change in the score (within the precision limits of integer arithmetic).

(4) A factor can have a special DON'T CARE value such that it has no effect on the score.

(5) The order of factor statements has no effect on the score.

(6) The total number of factors does not bias the score.

The first three of the properties relate to how individual factors influence the overall score. It should surprise no one that the score is high when all the factors are high. The score is low when any factor is very low, because a bad factor is a good clue that the system is on a false path. It is all right to blend together high factors, but a low factor deserves special attention. Currently, this is achieved by causing a bad factor to reduce the composite score in proportion to the degree that the factor falls below a certain threshold. Thus high factors combine additively to form an average, while low factors have a multiplicative effect that inhibits the entire result. In either case, an increase in any factor will produce an increase in the score, and conversely, a factor decrease yields a score decrease. This sensitivity is clearly desirable if the information from the factors is to be conveyed effectively to the parser.

While the first three properties have to do with how factors influence the score, the final three deal with ways that factors do not influence the score. First, there is provision for a DON'T CARE value so that a factor can leave the score unaffected in case it has no contribution to make with respect to evaluating a particular construct. Second, the score is independent of the order of factor statements so that the order can reflect the relative cost of evaluating the factor and its likelihood of producing a zero value that would free the system from evaluating the remainder of the factors. Finally, the number of factors does not bias the score either up or down. A phrase with 20 average value factors will not get a better or worse score than one with only 10, as would happen, for instance, if the factors were treated as independent probabilities and multiplied together.

The general outline of the algorithm is as follows:

```
Initialize WEIGHT = 0, TOTAL = 0, INHIB = 100;
For each factor F
    If F is NIL (the DON'T CARE value) then go on to next factor;
    If F is an integer then let W=1, T=F
    Otherwise F is <WEIGHT(F), TOTAL(F)>
        so let W=WEIGHT(F), T=TOTAL(F);
    Set WEIGHT to WEIGHT+W;
    Let Y = T/W;
```

If Y is greater than the threshold L then TOTAL = TOTAL+T

Otherwise TOTAL = TOTAL+W*L, and

INHIB = INHIB*Y/L;

Go on to next factor.

After all factors are completed, the resulting score is the pair <WEIGHT, TOTAL*INHIB/100>. The threshold L is set to 50 in the current version of the algorithm.

The result is left in the form of a pair, <WEIGHT, TOTAL>, instead of being reduced to a single integer, TOTAL/WEIGHT, so that constituent scores can make an appropriate contribution as factors in larger phrases. This is best illustrated through a simple example. Consider a hypothetical phrase P with a single constituent X and a total of four factors: one that comes from X's score and three others named A, B, and C. Let X's score in turn depend on three factors D, E, and F. The two cases to be considered are (1) scores represented by single integers and (2) scores represented by pairs. In the first case, assuming that all factors are above the threshold, the score of P is equal to

(A + B + C + D/3 + E/3 + F/3)/4.

In this case, the factors from the constituent are less important and the effect will be compounded with each further level of embedding. On the other hand, if scores are left as pairs then the score of P is

<6, A + B + C + D + E + F>.

This makes factors from constituents as important as higher level factors independent of level of embedding. This is an important property and, in fact, is the reason for having scores of the form <WEIGHT, TOTAL>.

5. Limitations of the Current Definition System

Before giving a formal description of the syntax of the language definition language, it is appropriate to discuss the limitations of the current definition system. While the SRI Speech Understanding project is not and need not be concerned with trying to produce a fully comprehensive language definition for English, it is important to consider what such an undertaking might require in evaluating the definition system and in contemplating extensions of it. The following paragraphs sketch a major source of problems for the current definition system and point out another problem area that actually seems to fall more in the domain of the parsing system.

In the present version of the definition system, and even in proposed versions including more powerful facilities for stating redundancy rules, the structure of the defined language is static in the sense that the possible immediate constituent patterns are all explicitly enumerated at the time the language is internalized. This is adequate for defining a large portion of a language such as English, but probably not for all of it. For certain constructions, it appears to be unreasonable to generate all the patterns ahead of time; instead, it may be necessary to have procedures for dynamically generating patterns so as to parse and understand the constructions. The distinguishing feature of these difficult cases is the juxtaposition of sentence fragments resulting from the deletion of a series of words that is not a constituent and that is duplicated (in a sense) somewhere else in the context.[4] The result often falls outside the standard patterns of the language and cannot be understood by the standard rules; before the construct can be parsed and understood, the deleted words must be accounted for and the appropriate constituent structure formed. Two major examples of this sort of process in English are comparative clauses and the various types of conjunctions.

The structure of comparative clauses is best explained as the result of both an obligatory deletion of some material that is identical to part of the head of the clause and an additional ellipsis of material that is identical to part of the higher clause containing the comparative (see Bresnan, 1973). Sentence 2 shows comparative deletion alone, and sentence 3 shows the result of comparative ellipsis.

------
[4] Not all deletions cause serious problems for the current system. For instance, the deletion occurring in relative clauses, while superficially similar, is much easier to deal with because it is limited to a single constituent.

(2) Dick told as many lies as John told.

(3) Dick told as many lies as John.

In sentence 4, deletions have resulted in a comparative clause that would not fall within the scope of the standard clause rules, as shown by sentence 5.

(4) He believes more of Dick's lies than he believes of John's.

(5) *He believes of John's.

Sentence 6 shows a comparative clause that fits the pattern of usual clauses, but it cannot be understood correctly according to the usual rules, as shown by the contrast with sentence 7.

(6) It is colder on the high road than it is on the low road.

(7) It is on the low road.

The comparative clause in sentence 4 must be reconstructed as something like "He believes x-many of John's lies" and the one in sentence 6 as "It is x-much cold on the low road." This reconstruction must be guided by the higher clause that dominates the comparative, and it is this extreme context dependency that makes comparative clauses a problem for the current definition system. Rather than a static set of rules for possible comparative clauses independent of context, a more adequate approach might be to include some sort of "active rule" as part of the language definition to reflect the processes by which comparatives are formed from standard clauses. Once internalized, the rule would operate during comprehension using the higher clause as a guide in dealing with the embedded comparative clause. Unfortunately, it is not yet clear how to formulate or internalize such a rule.

Another problem area that seems to call for definition in dynamic rather than static terms is the complex collection of phenomena labeled conjunction. Conjoined structures can show up in so many places and take so many forms that any attempt to define all the possibilities through a set of static rules seems clearly futile. A common form of conjunction, called gapping, will serve to illustrate some of the difficulties. In gapping, two or more clauses are conjoined and part of each secondary clause is deleted where it duplicates a portion of the first (see Ross, 1970). The deleted string is not limited to a single constituent as is shown in sentences 8 and 9.

(8) Dick could have easily been confused, and John misled.

(9) Dick needs to see a psychiatrist, and John a lawyer.

As with comparatives, 'gapped' clauses often violate the standard patterns of the language and are incomprehensible unless the deleted portion is accounted for. It appears that like comparatives, gapping would be best dealt with by an active rule--in this case, one that would use the first conjunct to guide the comprehension of the second. Other forms of conjunction present similar problems.

There has been little experimentation in AI and computational linguistics related to constructions as complicated as comparatives and conjunctions. The main exception is an experimental facility developed by Woods as part of a parsing system for transition network grammars (see Woods, 1973).[5] Woods' conjunction facility deals with sentences like 10 in which fragments are conjoined in a single clause with shared material factored out to the left and right (see Ross, 1967, pp. 97 ff.).

(10) Dick set his sights on and finally achieved complete ignominy.

The main shortcoming of Woods' special facility is that it is a special facility. Conjunction reduction, gapping, comparatives, and the like, should be dealt with through general mechanisms and as part of the language definition rather than by intricate modifications to the parsing system. Woods' experiment is important as an attempt to treat conjunction reduction in a parser, but it does not address the problem of stating such processes in a linguistically and computationally reasonable way as part of the language definition, or the problem of allowing several such processes to coexist as part of a single system. These problems are not going to be solved easily, but solutions must be found before understanding systems can deal with the full

------

[5] Another treatment of conjunction is found in Winograd's SHRDLU system (Winograd, 1971). However, his method seems to be best suited for dealing with conjunction of complete constituents, a much simpler type of construction than we are concerned with here.

complexity of natural language.

Another group of problems may ultimately lead to changes in the way the language is defined for the system. The common source of these problems is the fact that people produce and understand utterances that are in one way or another anomalous. Everyday discourse is filled with utterances containing false starts, unfinished phrases, "uh's", "um's", and a variety of other distortions.[6] Whether the result of a performance error by a competent speaker of the language or the lack of competence of a nonnative speaker, such utterances can often be understood by a human and will have to be equally comprehensible to a computer system that is expected to carry on a completely natural dialog. There is no debate about the existence of such utterances or the need to deal with them eventually; the issue is whether radical changes in the approach to defining the language will be needed.

Some have implied that the change in the way in which the language is defined must be very large indeed; for instance, either a change to a pattern-matching approach that will simply allow parts of the input to be ignored (Enea and Colby, 1973), or a change to a 'semantics-based' approach that refers to syntactic relationships only as a last resort (Schank and Tesler, 1969,

------

[6] See Chapanis (1975) for some examples of, as he puts it, "how untidy normal human conversations really are." Fromkin (1971) also provides examples and argues that anomalous utterances can provide insights into the organization underlying linguistic performance.

Riesbeck, 1974). An alternative that appears more attractive than these two is to deal with anomalies through mechanisms in the parsing system and leave the language definition intact. While techniques like loose pattern-matching and 'meaning-driven' analysis may be crucial for the parser to use in dealing with completely unconstrained dialog, it seems to be overreacting to conclude that they should form the standard mode of dealing with or defining language.

It appears reasonable, instead, to try a mixed-mode parsing strategy of first pushing as far as possible a structure-driven phase using the standard patterns of the language. If this produced only a collection of fragments, a meaning-driven phase would be entered to try to interpret the fragments as an anomalous, but perhaps comprehensible, message. Such a mixed-mode approach should have a reasonable chance of resulting in an efficient system that is still able to deal with errorful input. The local and relatively simple-to-process structural cues would be used to do as much of the job as possible, but if they failed to produce a complete parse, then more powerful--and more costly--conceptual analysis routines would be called on to try to make sense of whatever was produced in the first phase.[7]

------

[7] This argument does not imply that semantics is not an important factor in guiding the first phase when the system is looking for a parse within the usual structure of the language.

In such a system, the language definition would reflect constructions free of false-starts and other such anomalies; the burden of dealing with these would be placed on the parsing system instead. While this would make more radical changes to the current definition system unnecessary, some changes would undoubtedly be required. The definition would not have to indicate all the possible errors, but it would have to be flexible enough to allow the parser to deal with them. The exact details depend on the parser, but the modifications to the definition system promise to be less formidable than those required to deal with constructions like comparatives and conjunctions that occur in error-free utterances.

C. Syntax of the Language Definition Language

For completeness and as a summary, this section presents an annotated formal syntax of the language definition language. The syntax is described by means of an extended version of BNF notation (Backus, 1959). A BNF syntax rule has the form:

<p> ::= pattern

meaning that p-type constructions must conform to the given pattern. When there are several alternative patterns, the rule has the form:

<p> ::= pattern 1 | pattern 2 ... | pattern n

where pattern is a series of elements, each of which is one of the following:

ELEMENT	STANDS FOR
<q>	a q-type construct.
$x	zero or more occurrences of x.
(x1...xn)	the sequence x1 to xn.
(x1 |...| xn)	one of the alternative xi's.
[x1...xn]	optional sequence x1..xn.
any other symbol	an occurrence of the symbol.

In this formalism, a language definition has the following form:

```
<language definition> ::= LANGUAGE.DEF <declarations> <lexicon>
                                $<composition rule>
```

The declarations contain general information about the language and have the syntax given by the following set of rules:

```
<declarations> ::= $(<decl>;) END;
<decl> ::= CATEGORIES <idsequence> |
           ROOT CATEGORY <identifier> |
           AFFIXES <idsequence> |
           <decl function> <function spec> |
           ATTRIBUTES $<attr decl> ENDATTRS
<decl function> ::= RESPONSEFN | CATEGORYFN | RULEFN | WORDFN
<function spec> ::= <function name> |
                    LAMBDA ([<idsequence>]) <expression>
```

```
<attr decl> ::= <attr decl cats> (HAS | HAVE) <idsequence>

<attr decl cats> ::= ALL [EXCEPT <idsequence>] | <idsequence>

<idsequence> ::= <identifier> $(, <identifier>)
```

The lexicon contains the definitions for the basic units of the language, the words. The words are categorized, and for each lexical category there is a category definition and a word class definition. The lexical category definition specifies attributes and factors for occurrences of words in this category, the function to be used in combining factor values into a composite score, and the redundancy function to be used with the word definitions. The <cf command>'s, SCORE and RESCHEDULE, are used to control the parser and are discussed further in the sections on the language definition compiler and the parsing system.

```
<lexicon> ::= $(<lexical category def> | <word class def>)

<lexical category def> ::= CATEGORY.DEF <category name>
                                $(<cdecl>;) END;

<cdecl> ::= ATTRIBUTES <cattr> $(, <cattr>) |
            FACTORS <cfstat> $(, <cfstat>) |
            (SCORE | WORDFN)  <function spec>

<cattr> ::= <attribute name> = <expression>

<cfstat> ::= <factor name> = <expression> | <cf command>

<cf command> ::= (SCORE | RESCHEDULE) [IF <expression>]
```

The word class definition gives each word in the category and the attribute values shared by all instances of the word.

```
<word class def> ::= WORDS.DEF <category name> $<word def>
                      ENDWORDS;

<word def> ::= <lexical entry name> [<word attrs>];

<word attrs> ::= <word attr> $(, <word attr>)

<word attr> ::= <attribute name> = <attribute value>
```

The composition rules specify how words and phrases combine to form other phrases. The rule pattern gives the sequence of affixes, words, and phrases to be combined and the category of the resulting phrase. Other parts of the rule define attributes and factors, name the score function, or comment on examples.

```
<composition rule> ::= RULE.DEF <rule name> <rule pattern>
                          $(<rule part>;) END;

<rule pattern> ::= <category name> = [<prefix>] <constit>
                      $<constit> [<suffix>];

<prefix> ::= <affix name>-

<suffix> ::= -<affix name>

<constit> ::= <constit spec> [: <constit name>]

<constit spec> ::= <category name> | "<token name>

<rule part> ::= ATTRIBUTES <rule attr> $(, <rule attr>) |
                FACTORS <rule factor> $(, <rule factor>) |
                SCORE <function spec> |
                EXAMPLES <text not containing ";">
```

The rule pattern must have one or more constituents and may also specify affixes. If a constituent is not explicitly named, it is given the name from the <constit spec>. The <constit spec>

can be either a category name, indicating that the constituent must be a phrase of that category, or a token, in which case the <token name> must be identical to a <lexical entry name> for a word in the special category TOKEN, and the constituent must be an instance of that word.

The rule attribute statements either give an expression for calculating the attribute value or indicate that the value is to be copied from one of the constituents. A factor statement is either an expression for calculating a factor score or a (possibly conditional) command to the parser to calculate the composite score.

```
<rule attr> ::= <attribute name> = <expression> |
     <attribute name> ${, <attribute name>} FROM <constit name>
<rule factor> ::= <factor name> = <expression> |
                  SCORE [IF <expression>]
```

The language definition language is actually an extension of an existing programming language, the System Development Corporation's INFIX LISP. The syntax of <identifier>'s and <expression>'s in INFIX LISP is Algol-like and is not discussed here. When used in a lexical category definition, an expression may include references to word attributes by using the attribute name like a free variable. Similarly, in a composition rule an expression may reference an attribute defined in the rule by its attribute name. Also, it may reference an attribute of a constituent by a subexpression of the form A(C) where A is the attribute name and C is the constituent name. Finally, in rule factor statements, the score of a constituent, C, may be referenced by SCORE(C).

D. Language Definition Compiler and Internal Representation

The role of the compiler in the definition system is to translate the external representation of the language into an internal form for use by the parser. The translation has three steps: external representation to first intermediate, first to second intermediate, and, finally second intermediate to internal. The first intermediate representation is a list structure containing the same information as the external form but formatted for easy manipulation by programs rather than for humans. The second intermediate representation has the same general structure as the first, but includes the changes made by the redundancy functions. The internal form is a complex representation anticipating the various ways in which the information will be used by the parsing system.

1. Category and Rule Records

The major components of the internal representation are records defining categories and composition rules. For each category in the language, there is a category record containing the following information:

(1) Attribute symbol table--used to convert names of attributes defined for the category into unique numeric indices.

(2) Word list--each word in the category is represented in the list by its lexical entry name and an attribute value array.

(3) Score function--provided in the category definition and used for words in this category to convert factor values into a score.

(4) Composition rules--a list of composition rule records for all the rules that construct phrases of this category.

(5) Rule occurrences--lists the rules and pattern positions where this category appears as a constituent specification.

(6) Focus tables--reflect possible constituent structures of the category for use in determining conflicts with the parser's focus of activity (see the section on The Parsing System for explanations of focus of activity, focus conflict, and the use of these tables).

(7) Factor function--a LISP function created from attribute and factor statements of the category definition and called by the parser to check for predicted words (details given below).

(8) Miscellaneous information such as the name of the

category and the names of factors.

The internal representation of a composition rule is a record holding the following:

(1) Category--pointer to a record representing the category of phrases produced by this rule.

(2) Affixes--affix names, if any, for prefix and suffix.

(3) Constituent pattern--list of constituent name/constituent specification pairs.

(4) Score function--provided in the rule definition and used to combine factor values into a composite score.

(5) Factor function--a LISP function created from attribute and factor statements and called by the parser to construct phrases according to this rule (discussed below).

(6) Miscellaneous information such as the name of the rule and the names of factors.

2. Factor Functions

The most complex component of both category and rule representations is the factor function. The language definition compiler converts the information in the attribute and factor statements into a LISP function that can be called during the parsing process. The function takes advantage of detailed knowledge of the data structures and run-time variables used by the parser, and, once compiled by the LISP compiler, is an efficient form in which to represent the attribute and factor specifications.

The function is constructed so that factors are evaluated in the same order as they are listed in the second intermediate representation. If any factor evaluates to zero, the rest are skipped. Attributes are evaluated as they are needed for the evaluation of factors, or at the end if no factor references them. The factor commands SCORE and RESCHEDULE receive special treatment. SCORE means call the score function with as many factors as have been evaluated and confirm that the resulting score is above a certain threshold. If it is not, the function terminates without evaluating any more attributes or factors. This action may be worthwhile before a costly attribute and factor combination such as the one that creates and tests the semantic representation. There is always an implicit SCORE at the end of the factors. The command RESCHEDULE can be used as a lexical factor statement; it means first calculate the score and then reschedule further processing on this word. The details of this operation, such as how to determine the priority at which the further processing is rescheduled, is discussed in the section on the parsing system. Both SCORE and RESCHEDULE can be conditional on the outcome of some test.[8]

------
[8] Rules often make SCOREing dependent on the parser variable VIRTUAL being NIL, indicating that a permanent phrase is to be constructed rather than a temporary, 'virtual' phrase. See the discussion of calculating phrase values in the section on the parsing system for more about virtual phrases.

As indicated in the description of the formal syntax of the language definition language, the expressions for factor and attribute statements are written in an extended version of SDC INFIX LISP. After conversion to a prefix form in the intermediate representation, the expressions are further processed before inclusion in the factor function. Attribute names are converted to references to particular elements of the attribute value array. For composition rules, references to constituent attributes are converted to forms that access the corresponding item from the constituent attribute array. All attribute and factor names are replaced in the factor functions by array accesses using numeric indices.

To illustrate the operation of the language definition compiler, the construction of sample factor functions is sketched for both a lexical category definition and a composition rule. This also provides an opportunity to show the changes that take place during the representation conversion from external to first intermediate, then to second intermediate, and finally to internal. The lexical definition is the simpler of the two and will be treated first. Figure II-4 gives the external representation of the lexical category definition for category NP. There is a single attribute statement to compute the SEMANTICS from the WDSEMANTICS attribute of the lexical entry. The two factor statements are simply a constant factor followed by an unconditional RESCHEDULE command.

Figure II-4 External Representation for a Category Definition

```
CATEGORY.DEF NP
    ATTRIBUTES   SEMANTICS = SEMCALL("SEMRNP5,WDSEMANTICS);
    FACTORS   INIT = 80, RESCHEDULE;
  END;
```

Figure II-5 contains the first intermediate representation of the category definition. The same information is present but reorganized and put in a list structure for further processing. The intermediate representation for a lexical definition is a five-tuple: category name, attribute specifications list, factor specifications list, score function, and word redundancy function. Each entry on the attribute specifications list is an attribute name followed by an expression to compute the attribute value. Similarly, the entries on the factor specifications list are name-expression pairs or factor commands, SCORE or RESCHEDULE, optionally followed by a test expression. The NILs for score function and word redundancy function simply indicate that these were not specified in the external representation.

Figure II-5 First Intermediate Representation
for a Category Definition

```
(NP ((SEMANTICS (SEMCALL (QUOTE SEMRNP5) WDSEMANTICS)))
    ((INIT 80)
     (RESCHEDULE))
    NIL NIL)
```

The second intermediate representation is given in Figure II-6. The redundancy function for category definitions has added three attributes and a factor, all related to matching the proposed word to the input signal. The MAPINFO attribute is set by calling the MAPPING function with the SPELLING of the word and the proposed position in the input given by the parser variables PLEFT and PRIGHT. The value of MAPINFO will be a list of the left word boundary, the right word boundary, and a score indicating the degree of match. The first two elements of this list determine the LEFT and RIGHT attributes, respectively, and the third element, the score, is passed to the MAPCNVT function along with the STRING attribute of the word to yield the MAPPING factor. Finally, the redundancy function has specified WORDSCOREFN as the score function for the category but has left the word redundancy function NIL.

Figure II-6 Second Intermediate Representation for a Category Definition

```
(NP ((SEMANTICS (SEMCALL (QUOTE SEMRNP5) WDSEMANTICS))
     (MAPINFO (MAPPING SPELLING PLEFT PRIGHT))
     (LEFT (CAR MAPINFO))
     (RIGHT (CADR MAPINFO)))
    ((INIT 80)
     (RESCHEDULE)
     (MAPPING (MAPCNVT (CADDR MAPINFO) STRING)))
    WORDSCOREFN
    NIL)
```

Figure II-7 gives the LISP factor function created by the language definition compiler from the intermediate definition in Figure II-6. It is not necessary to go into all the details of the function definition to make the most important points. The first of these is that there is extensive dependency on details of the operation of the parser in the form of calls on parser functions, references to parser variables, and direct manipulation of parser data structures. The second point is that there is a large increase in complexity relative to the earlier representations due both to the intricate relationship to the parser and to the explicit presence of a variety of items such as control statements, zero tests for factors, score calculation, score threshold tests, list manipulation to save factor values, and array manipulation to record and access attribute values. The contrast between the original definition in Figure II-4 and the factor function in Figure II-7, which is only one component of the internal category definition for use by the parser, shows the importance of separate external and internal representations and the need for automatic compilation of the internal from the external.

The translation from external to internal representation is even more striking for composition rules. Figure II-8 contains a rule definition taken from the SRI language definition. (It is the full form of the rule used as an example earlier). The first intermediate form of the rule is given in Figure II-9. An intermediate rule representation is a five-tuple consisting of rule name, rule pattern, attribute specifications list, factor specifications list, and score function. The rule pattern is a list of category name, prefix, suffix, and constituents. Each constituent is given as a name and constituent specification pair. The attribute and factor specifications lists and the score function are the same as for lexical category definitions.

Figure II-7 LISP Factor Function for a Category Definition

```
(NP.LEXFACTORFN (LAMBDA (CFALTWORD)
  (PROG ((CFALTATTRS (ROR (CADR CFALTWORD) (CADAR CFALTWORD)))
     FACTV U V W X Y Z)
  (CASEGO (LENGTH (CDR CFALTWORD)) L1 L2 FIN)
  L1
  (NCONC CFALTWORD (CONS 80 NIL))
  (SETQ CFALTSCORE (APPLYX SCOREF CFALTSCORE (CDDR CFALTWORD)))
  (COND ((SLQ (SCR2INT CFALTSCORE) CTPRUNETHRESHHOLD) (GO PRUNE)))
  (RETURN (QUOTE RESCHEDULE))
  L2
  (RPLACA (CDR CFALTWORD) (SETQ CFALTATTRS (COPYPRSARRAY
     (CADAR CFALTWORD))))
  (SETA CFALTATTRS 23 (MAPPING (GETA CFALTATTRS 22) PLEFT PRIGHT))
  (COND ((EQ 0 (SETQ FACTV
     (MAPCNVT (CADDR (GETA CFALTATTRS 23))
     (GETA CFALTATTRS 3))))) (GO PRUNE)))
  (NCONC CFALTWORD (CONS FACTV NIL))
  FIN
  (SETQ CFALTSCORE (APPLYX SCOREF CFALTSCORE (CDDR CFALTWORD)))
  (COND ((SLQ (SCR2INT CFALTSCORE) CTPRUNETHRESHHOLD) (GO PRUNE)))
  (SETA CFALTATTRS 16 (SEMCALL (QUOTE SEMRNP5)
     (GETA CFALTATTRS 15)))
  (SETA CFALTATTRS 1 (CAR (GETA CFALTATTRS 23)))
  (SETA CFALTATTRS 2 (CADR (GETA CFALTATTRS 23)))
  (RETURN (QUOTE SPAWN))
  PRUNE
  (RETURN (QUOTE PRUNE]
```

In Figure II-10, the second intermediate representation is shown after the rule redundancy function has been applied to increase the number of attributes from 8 to 17, increase the number of factor statements from 13 to 26, and specify the score function. Finally, the immense factor function produced by the compiler is presented in Figure II-11. Like the lexical factor function, this one reflects detailed knowledge of the parser design and is much more complex and difficult to comprehend than the external representation. These would be critical defects if humans had to deal with factor functions directly; however, since the functions are constructed automatically and never seen by the researchers (except the ones debugging the language definition compiler), what would be defects can be accepted as harmless side effects of the desire for efficiency.

Figure II-8 External Representation for a Composition Rule Definition

```
RULE.DEF S8    S = AUXB NP:NP1 NP:NP2;
  ATTRIBUTES
    RELN,CMU,FOCUS FROM NP1,
    MOOD = "(YN),
    TRANS = 0,
    AFFNEG FROM AUXB,
    SEMANTICS = SEMCALL("SEMRS8,SEMANTICS(NP1),SEMANTICS(NP2)),
    PITCHC = FINDPITCHC(PLEFT,PRIGHT);
  FACTORS
    GCASE1 = IF GCASE(NP1) EQUAL "(ACC) THEN OUT ELSE OK,
    PROB = LK1,
    GCASE2 = IF GCASE(NP2) EQUAL "(ACC) THEN OUT ELSE OK,
    MOOD1 = IF MOOD(NP1) EQUAL "(WH) THEN BAD ELSE OK,
    MOOD2 = IF MOOD(NP2) EQUAL "(WH) THEN BAD ELSE OK,
    NBRAGR1   IF CMU EQUAL "(UNIT) THEN
      [IF NBR(AUXB) EQUAL "(SG) THEN OK ELSE OUT]
      ELSE IF GINTERSECT(NBR(NP1),NBR(NP2)) THEN OK ELSE OUT,
    NBRAGR2 = IF CMU(NP2) EQUAL "(UNIT) THEN OK ELSE
      IF GINTERSECT(NBR(NP2),NBR(AUXB)) THEN OK ELSE OUT,
    PERSAGR = IF GINTERSECT(PERS(NP1),PERS(AUXB))
      THEN OK ELSE OUT,
    FOCUS = IF FOCUS(NP1) EQ "INDEF AND FOCUS(NP2) EQ "DEF
      THEN POOR ELSE OK,
    RELN = IF RELN EQ "T THEN
      IF CMU EQUAL "(UNIT) THEN VERYGOOD ELSE OK,
    SCORE IF NOT VIRTUAL,
    STRESS = IF VIRTUAL THEN OK ELSE
      SELECTQ STRESS(AUXB) WHEN UNREDUCED THEN GOOD,
    PITCHC = IF VIRTUAL THEN OK ELSE
      IF PITCHC EQ "HIRISE THEN GOOD ELSE OK;
  EXAMPLES
    IS A LAFAYETTE THE SUBMARINE? (POOR)
    IS IT A LAFAYETTE??(GOOD,I.E. WITH HIRISE)
    IS WHAT THE SURFACE DISPLACEMENT (BAD),
    IS THE LAFAYETTE A SUBMARINE? (OK);
  END;
```

Figure II-9 First Intermediate Representation
for a Composition Rule Definition

```
(S8 (S NIL NIL (AUXB AUXB) (NP1 NP) (NP2 NP))
    ((RELN (RELN NP1))
     (CMU (CMU NP1))
     (FOCUS (FOCUS NP1))
     (MOOD (QUOTE (YN)))
     (TRANS 0)
     (AFFNEG (AFFNEG AUXB))
     (SEMANTICS (SEMCALL (QUOTE SEMRS8) (SEMANTICS NP1)
                (SEMANTICS NP2)))
     (PITCHC (FINDPITCHC PLEFT PRIGHT)))
    ((GCASE1 (COND ((EQUAL (GCASE NP1) (QUOTE (ACC))) OUT)
             (T OK)))
     (PROB LK1)
     (GCASE2 (COND ((EQUAL (GCASE NP2) (QUOTE (ACC))) OUT)
             (T OK)))
     (MOOD1 (COND ((EQUAL (MOOD NP1) (QUOTE (WH))) BAD) (T OK)))
     (MOOD2 (COND ((EQUAL (MOOD NP2) (QUOTE (WH))) BAD) (T OK)))
     (NBRAGR1 (COND ((EQUAL CMU (QUOTE (UNIT)))
                     (PROGN (COND ((EQUAL (NBR AUXB)
                                  (QUOTE (SG))) OK) (T OUT))))
                    (T (COND ((GINTERSECT (NBR NP1) (NBR NP2)) OK)
                                  (T OUT)))))
     (NBRAGR2 (COND ((EQUAL (CMU NP2) (QUOTE (UNIT))) OK)
                    (T (COND ((GINTERSECT (NBR NP2)
                    (NBR AUXB)) OK) (T OUT)))))
     (PERSAGR (COND ((GINTERSECT (PERS NP1) (PERS AUXB)) OK)
                    (T OUT)))
     (FOCUS (COND ((AND (EQ (FOCUS NP1) (QUOTE INDEF))
                        (EQ (FOCUS NP2) (QUOTE DEF))) POOR)
                  (T OK)))
     (RELN (COND ((EQ RELN (QUOTE T))
                  (COND ((EQUAL CMU (QUOTE (UNIT))) VERYGOOD)
                                (T OK)))))
     (SCORE (NOT VIRTUAL))
     (STRESS (COND (VIRTUAL OK)
                   (T (SELECTQ (STRESS AUXB) (UNREDUCED GOOD)
                                NIL))))
     (PITCHC (COND (VIRTUAL OK)
                   (T (COND ((EQ PITCHC (QUOTE HIRISE)) GOOD)
                                (T OK))))))
    NIL)
```

Figure II-10 Second Intermediate Representation for a Composition Rule Definition

```
(S8 (S NIL NIL (AUXB AUXB) (NP1 NP) (NP2 NP))
    ((PHRMAPINFO (PHRM STRING PLEFT PRIGHT))
     (LSTWD (PROGN (SETQ X STRING)
                   (COND ((OR (EQ X (QUOTE UNDEFINED))
                              (NULL (LASTEL X)))
                          (QUOTE UNDEFINED))
                         (T (LASTEL X)))))
     (FSTWD (PROGN (SETQ X STRING)
                   (COND ((OR (EQ X (QUOTE UNDEFINED))
                              (NULL (CAR X)))
                          (QUOTE UNDEFINED))
                         (T (CAR X)))))
     (STRING (APPENDALL (STRING AUXB) (STRING NP1) (STRING NP2)))
     (BULK (ADDBULK 2 (BULK AUXB) 2 (BULK NP1) 2 (BULK NP2) 2))
     (DEPTH (MAXDEPTH (DEPTH AUXB) 2 (DEPTH NP1) 2 (DEPTH NP2) 2))
     (SIZE (ADDSIZE (SIZE AUXB) (SIZE NP1) (SIZE NP2)))
     (RIGHT (SETRIGHT (RIGHT NP2) PHRMAPINFO))
     (LEFT (SETLEFT (LEFT AUXB) PHRMAPINFO))
     (RELN (RELN NP1))
     (CMU (CMU NP1))
     (FOCUS (FOCUS NP1))
     (MOOD (QUOTE (YN)))
     (TRANS 0)
     (AFFNEG (AFFNEG AUXB))
     (SEMANTICS (SEMCALL (QUOTE SEMRS8) (SEMANTICS NP1)
                (SEMANTICS NP2)))
     (PITCHC (FINDPITCHC PLEFT PRIGHT)))
    ((NP2 (CSCORE (SCORE NP2)))
     (NP1 (CSCORE (SCORE NP1)))
     (AUXB (CSCORE (SCORE AUXB)))
     (BOTHFIXED (CHECKTIMES LEFT RIGHT))
     (GCASE1 (COND ((EQUAL (GCASE NP1) (QUOTE (ACC))) OUT)
                   (T OK)))
     (PROB LK1)
     (GCASE2 (COND ((EQUAL (GCASE NP2) (QUOTE (ACC))) OUT)
                   (T OK)))
     (MOOD1 (COND ((EQUAL (MOOD NP1) (QUOTE (WH))) BAD) (T OK)))
     (MOOD2 (COND ((EQUAL (MOOD NP2) (QUOTE (WH))) BAD) (T OK)))
     (NBRAGR1 (COND ((EQUAL CMU (QUOTE (UNIT)))
                     (PROGN (COND ((EQUAL (NBR AUXB)
                                          (QUOTE (SG))) OK) (T OUT))))
                    (T (COND ((GINTERSECT (NBR NP1) (NBR NP2)) OK)
                             T OUT)))))
```

Figure II-10 Second Intermediate Representation for a Composition Rule Definition (concluded)

```
(NBRAGR2 (COND ((EQUAL (CMU NP2) (QUOTE (UNIT))) OK)
               (T (COND ((GINTERSECT (NBR NP2)
                                     (NBR AUXB)) OK)
                        (T OUT)))))
(PERSAGR (COND ((GINTERSECT (PERS NP1) (PERS AUXB)) OK)
                      (T OUT)))
(FOCUS (COND ((AND (EQ (FOCUS NP1) (QUOTE INDEF))
                   (EQ (FOCUS NP2) (QUOTE DEF))) POOR)
             (T OK)))
(RELN (COND ((EQ RELN (QUOTE T))
             (COND ((EQUAL CMU (QUOTE (UNIT))) VERYGOOD)
                      (T OK)))))
(SCORE (NOT VIRTUAL))
(STRESS (COND (VIRTUAL OK)
              (T (SELECTQ (STRESS AUXB)
                          (UNREDUCED GOOD) NIL))))
(PITCHC (COND (VIRTUAL OK)
              (T (COND ((EQ PITCHC (QUOTE HIRISE)) GOOD)
                      (T OK)))))
(DEPTH (DEPTHSCORE DEPTH))
(BULK (BULKSCORE BULK))
(SCORE (NOT VIRTUAL))
(PHRMAPPING (COND (VIRTUAL OK)
                  (T (PMCHECK PHRMAPINFO STRING))))
(SCORE (NOT VIRTUAL))
(COART (COND (VIRTUAL OK) (T (COART (RIGHT AUXB)
                                    (LEFT NP1)))))
(COART (COND (VIRTUAL OK) (T (COART (RIGHT NP1)
                                    (LEFT NP2)))))
(SCORE (NOT VIRTUAL))
(SEMANTICS (SEMCHK SEMANTICS)))
RULESCOREFN)
```

Figure II-11 Factor Function for a Composition Rule Definition

```
(S8.FACTORFN (LAMBDA NIL (PROG (U V W X Y Z)
    (COND ((EQ 0 (SETA RFFACTORVALS 1 (CSCORE
                    (NTHEL RFRHSSCORES 3)))) (GO FAIL)))
    (COND ((EQ 0 (SETA RFFACTORVALS 2 (CSCORE
                    (NTHEL RFRHSSCORES 2)))) (GO FAIL)))
    (COND ((EQ 0 (SETA RFFACTORVALS 3 (CSCORE
                    (NTHEL RFRHSSCORES 1)))) (GO FAIL)))
    (SETA RFATTRS 3 (APPENDALL (GETA RFC1ATTRS 3)
                          (GETA RFC2ATTRS 3)) (GETA RFC3ATTRS 3)))
    (SETA RFATTRS 20 (PHRM (GETA RFATTRS 3) PLEFT PRIGHT))
    (SETA RFATTRS 2 (SETRIGHT (GETA RFC3ATTRS 2)
                                (GETA RFATTRS 20)))
    (SETA RFATTRS 1 (SETLEFT (GETA RFC1ATTRS 1)
                               (GETA RFATTRS 20)))
    (COND ((EQ 0 (SETA RFFACTORVALS 4
                 (CHECKTIMES (GETA RFATTRS 1) (GETA RFATTRS 2))))
           (GO FAIL)))
    (COND ((EQ 0 (SETA RFFACTORVALS 5
            (COND ((EQUAL (GETA RFC2ATTRS 11) (QUOTE (ACC))) OUT)
                          (T OK))))
         (GO FAIL)))
    (COND ((EQ 0 (SETA RFFACTORVALS 6 LK1)) (GO FAIL)))
    (COND ((EQ 0 (SETA RFFACTORVALS 7
            (COND ((EQUAL (GETA RFC3ATTRS 11) (QUOTE (ACC))) OUT)
                          (T OK)))) (GO FAIL)))
    (COND ((EQ 0 (SETA RFFACTORVALS 8
            (COND ((EQUAL (GETA RFC2ATTRS 14) (QUOTE (WH))) BAD)
                             (T OK)))) (GO FAIL)))
    (COND ((EQ 0 (SETA RFFACTORVALS 9
            (COND ((EQUAL (GETA RFC3ATTRS 14) (QUOTE (WH))) BAD)
                             (T OK)))) (GO FAIL)))
    (SETA RFATTRS 5 (GETA RFC2ATTRS 7))
    (COND ((EQ 0 (SETA RFFACTORVALS 10
            (COND ((EQUAL (GETA RFATTRS 5) (QUOTE (UNIT)))
                     (PROGN (COND ((EQUAL (GETA RFC1ATTRS 6)
                                    (QUOTE (SG))) OK) (T OUT))))
                   (T (COND ((GINTERSECT (GETA RFC2ATTRS 12)
                                  (GETA RFC3ATTRS 12)) OK)
                          (T OUT)))))) (GO FAIL)))
    (COND ((EQ 0 (SETA RFFACTORVALS 11
           (COND ((EQUAL (GETA RFC3ATTRS 7) (QUOTE (UNIT))) OK)
                 (T (COND ((GINTERSECT (GETA RFC3ATTRS 12)
                                (GETA RFC1ATTRS 6)) OK)
                        (T OUT)))))) (GO FAIL)))
```

Figure II-11 Factor Function for a Composition Rule Definition
(continued)

```
(COND ((EQ 0 (SETA RFFACTORVALS 12
            (COND ((GINTERSECT (GETA RFC2ATTRS 9)
                               (GETA RFC1ATTRS 4)) OK)
                  (T OUT)))) (GO FAIL)))
(COND ((EQ 0 (SETA RFFACTORVALS 13
        (COND ((AND (EQ (GETA RFC2ATTRS 10) (QUOTE INDEF))
                    (EQ (GETA RFC3ATTRS 10) (QUOTE DEF))) POOR)
                   (T OK)))) (GO FAIL)))
(SETA RFATTRS 4 (GETA RFC2ATTRS 5))
(COND ((EQ 0 (SETA RFFACTORVALS 14
            (COND ((EQ (GETA RFATTRS 4) (QUOTE T))
                   (COND ((EQUAL (GETA RFATTRS 5) (QUOTE (UNIT)))
                          VERYGOOD) (T OK)))))) (GO FAIL)))
(COND ((NOT VIRTUAL)
        (SETQ RFSCORE (APPLYX RFSCOREFN RFSCORE RFFACTORVALS))
         (COND ((SLQ (SCR2INT RFSCORE) CTPRUNETHRESHHOLD)
         (GO FAIL)))))
(COND ((EQ 0 (SETA RFFACTORVALS 15
                 (COND (VIRTUAL OK)
                       (T (SELECTQ (GETA RFC1ATTRS 5)
                            (UNREDUCED GOOD) NIL))))) (GO FAIL)))
(SETA RFATTRS 6 (FINDPITCHC PLEFT PRIGHT))
(COND ((EQ 0 (SETA  RFFACTORVALS 16
        (COND (VIRTUAL OK)
           (T (COND ((EQ (GETA RFATTRS 6) (QUOTE HIRISE)) GOOD)
                                (T OK)))))) (GO FAIL)))
(SETA RFATTRS 14 (MAXDEPTH (GETA RFC1ATTRS 10)
         2 (GETA RFC2ATTRS 20) 2 (GETA RFC3ATTRS 20) 2))
(COND ((EQ 0 (SETA RFFACTORVALS 17
         (DEPTHSCORE (GETA RFATTRS 14)))) (GO FAIL)))
(SETA RFATTRS 15 (ADDBULK 2 (GETA RFC1ATTRS 11)
         2 (GETA RFC2ATTRS 21) 2 (GETA RFC3ATTRS 21) 2))
(COND ((EQ 0 (SETA RFFACTORVALS 18 (BULKSCORE
         (GETA RFATTRS 15)))) (GO FAIL)))
(COND ((NOT VIRTUAL)
        (SETQ RFSCORE (APPLYX RFSCOREFN RFSCORE RFFACTORVALS))
(COND ((SLQ (SCR2INT RFSCORE) CTPRUNETHRESHHOLD) (GO FAIL)))))
(COND ((EQ 0 (SETA RFFACTORVALS 19
         (COND (VIRTUAL OK)
               (T (PMCHECK (GETA RFATTRS 20)
                           (GETA RFATTRS 3)))))) (GO FAIL)))
(COND ((NOT VIRTUAL)
        (SETQ RFSCORE (APPLYX  RFSCOREFN RFSCORE RFFACTORVALS))
(COND ((SLQ (SCR2INT RFSCORE) CTPRUNETHRESHHOLD) (GO FAIL)))))
```

Figure II-11 Factor Function for a Composition Rule Definition (concluded)

```
(COND ((EQ 0 (SETA RFFACTORVALS 20
      (COND (VIRTUAL OK)
        (T (COART (GETA RFC1ATTRS 2) (GETA RFC2ATTRS 1))))))
   (GO FAIL)))
(COND ((EQ 0 (SETA RFFACTORVALS 21
          (COND (VIRTUAL OK)
               (T (COART (GETA RFC2ATTRS 2)
                         (GETA RFC3ATTRS 1)))))) (GO FAIL)))
(COND ((NOT VIRTUAL)
     (SETQ RFSCORE (APPLYX RFSCOREFN RFSCORE RFFACTORVALS))
(COND ((SLQ (SCR2INT RFSCORE) CTPRUNETHRESHHOLD) (GO FAIL)))))
(SETA RFATTRS 12 (SEMCALL (QUOTE SEMRS8) (GETA RFC2ATTRS 18)
                            (GETA RFC3ATTRS 18)))
(COND ((EQ 0 (SETA RFFACTORVALS 22
                (SEMCHK (GETA RFATTRS 12)))) (GO FAIL)))
(SETQ RFSCORE (APPLYX RFSCOREFN RFSCORE RFFACTORVALS))
(COND ((SLQ (SCR2INT RFSCORE) CTPRUNETHRESHHOLD) (GO FAIL)))
(SETA RFATTRS 17 (PROGN (SETQ X (GETA RFATTRS 3))
      (COND ((OR (EQ X (QUOTE UNDEFINED)) (NULL (LASTEL X)))
               (QUOTE UNDEFINED))
              (T (LASTEL X)))))
(SETA RFATTRS 16 (PROGN (SETQ X (GETA RFATTRS 3))
          (COND ((OR (EQ X (QUOTE UNDEFINED)) (NULL (CAR X)))
                 (QUOTE UNDEFINED))
               (T (CAR X)))))
(SETA RFATTRS 13 (ADDSIZE (GETA RFC1ATTRS 9)
              (GETA RFC2ATTRS 19) (GETA RFC3ATTRS 19)))
(SETA RFATTRS 7 (GETA RFC2ATTRS 10))
(SETA RFATTRS 9 (QUOTE (YN)))
(SETA RFATTRS 8 0)
(SETA RFATTRS 10 (GETA RFC1ATTRS 7))
(RETURN T)
FAIL (RETURN NIL))))
```

E. Conclusions

The most significant features of the definition system are the prominent place given to factors for evaluating phrases, the emphasis on different definition representations for human and computer, and the first steps toward a capability for including generalizations about the language in the form of redundancy rules. The factor mechanism provides a uniform way of integrating a variety of knowledge sources, many of which may depend on uncertain information or probabilistic tendencies. As such, factors are of practical interest as an approach to problems of system integration and guidance of the parsing process. These issues are discussed elsewhere in relation to the parser. In addition, factors may be of interest linguistically with respect to work on systematic covariation (modeled by Labov and others with the aid of 'variable rules'; see Cedergren and Sankoff, 1974) and work on quasi-continuous, 'squishy' phenomena in language (work begun and most intensively pursued by Ross; see Ross, 1972, 1973a, 1973b, and Lakoff, 1973).

The use of different representations and automatic compilation of the computer's internal form of the language definition from the human's external form of the definition have several beneficial results. The most important is the increased freedom in the attempt to satisfy jointly the conflicting goals of having a representation that leads to efficient computation while also having one that allows a clear definition of the language.

An additional benefit of the dual representation approach, is the ability to make the external form of the definition relatively neutral with respect to the design of the parser while still having an internal representation tailor-made for the particular parsing strategy. Furthermore, changes in the representational needs of the parsing system can often be accommodated by making changes in the language definition compiler rather than modifying the definition itself.

Redundancy rules are expected to be of increasing importance as a way of stating generalizations that will simplify the language definition. With redundancy rules that are applied during the compilation process, it should be possible to state in a single place in the external definition a generalization about the language that has widespread effects on the internal definition. Thus the information about some language feature can be concentrated in one place in the human's version while still being given whatever is found to be the most efficient representation in the computer's version.

The development of the definition system has been influenced by three main sources of ideas: the work in linguistics on various approaches to defining natural languages, the work in computer science on translator writing systems for programming languages (see Feldman and Gries, 1968), and the other work in artificial intelligence on language understanding. In this last category, one influence on the development of the definition system was a series of discussions, not always ending in agreement, with the members of the PHLIQA project of Philips Research Laboratories.[9] Their stalwart defense of the use of restricted context free rules and the value of distinguishing formal definition from implementation details must have contributed to our own shift away from a strict 'proceduralist' view (Paxton and Robinson, 1973; Paxton, 1974).

The definition system (and the complementary parsing system described below) is written in SDC INFIX LISP and runs in the SDC LISP system on the IBM 370 and (through a translator) in INTERLISP on the DEC PDP 10. It is structured so that, in addition to being able to compile an entire language definition, parts of the definition can be individually recompiled. For example, if an attribute or factor statement in a particular rule is changed, the internal language definition can be updated by simply recompiling that one rule. This is a valuable capability with a definition that is undergoing continual refinement and development.

There are two major forces for change in the definition system: human-motivated demands for extensions to the external representation and computer-motivated demands for revisions of the internal one. In the former case, advances hoped for in the utilization of redundancy rules are both the development of definition language forms for defining the rules and research into

------
[9] No reports have yet been published by the Philips group.

rules with global effects on the structure of the language. Research will also be needed regarding the 'active' rules referred to in the previous discussion of limitations of the current system. In the case of changes prompted by the needs of the parsing system, more experimentation is necessary to determine what further modifications will be required. The experience to date has been that revised parser demands can usually be satisfied by changes to the compiler without affecting the external form of the language definition. Whether this satisfying trend continues depends largely on the currently hard-to-predict evolution of the parsing system described in Section III.

III THE PARSING SYSTEM

Prepared by William H. Paxton

Contents:

A. Introduction

The activity of the parsing system can be described as the step by step construction of 'interpretations' of utterances. An interpretation is a phrase of the root category of the language that spans the utterance and includes attributes such as semantic representation. Phrases are created by either (1) recognizing a word in the input or (2) applying a composition rule to constituent phrases. In the parser's search for an appropriate interpretation, phrases are incrementally formed, evaluated, and combined. As this process goes on, the parser builds a data structure, called the 'parse net', representing the growing collection of phrases, and maintains another structure, called the 'task queue', encoding the alternative operations available for taking another step toward understanding the input. Each entry in the task queue specifies a procedure to be performed at a particular location (node) in the parse net. The performance of such a procedure typically entails both modifying the parse net and scheduling new tasks to make further modifications. By factoring the parsing process into tasks that first make incremental changes and then spawn other tasks to be performed at unspecified later times, the parser is given a means of controlling the overall activity of the understanding system. Other components of the system such as semantics and acoustics may carry out large portions of a task, but it is the responsibility of the parser to decide when the task will actually be performed. Thus instead of having a separate 'control' component in the system, decisions regarding what to do next are made by the parser on the basis of a complex, heuristic parsing strategy described at length below.

The control aspect of the parser's role is of great importance, because only a subset of the scheduled tasks will actually prove to be necessary to understand the input; the others will be 'false steps' leading toward potential interpretations but proving to be inappropriate for the particular utterance being parsed. Ideally, in deciding which task to perform next, the parser would always choose one of the necessary tasks and never take a false step. The utterance would be understood with the unnecessary tasks still left in the queue. To approach this ideal, the actual system must spend some of its effort deciding which task to perform next. Such effort is well spent if it produces a net decrease in processing time. In other words, the efficiency of the system will be improved by decisions regarding the order in which tasks are performed if the cost of the decisions is less than the cost of the false step tasks that would have otherwise been performed. Since the potential for wasting effort on unnecessary operations is particularly large in speech understanding, the system can afford to carry out rather complex computations in deciding what to do next, and still get a big improvement in overall efficiency. In the current system, the decisions are based on the relative priorities assigned to the various tasks waiting in the queue.

In establishing priorities, the parser gets important guidance from the 'values' the language definition assigns to different interpretations. Recall that in addition to defining the possible phrases, the language definition also associates with each phrase a set of factors to be used in establishing its score with respect to particular input signals and contexts.[1] In

------
[1] See the discussion of factors and scores in Section II, The Definition System.

particular, each interpretation, being a root category phrase, gets a score in this manner. The interpretation value is a simple function (given below) of this root score. Other things being equal, a task will be favored if it appears to lead toward an interpretation with a higher value. To achieve this ranking, task priorities assigned by the parser tend to reflect the maximum value of the interpretations whose construction the task would lead to.

In addition to interpretation value, response time is also an important concern. The parser must balance the goal of finding the interpretation with the highest value against the goal of making a prompt response. Our approach to dealing with these conflicting goals is to maintain in the parser a set of phrases, called 'focus phrases', that have been constructed in the parse and to concentrate on finding ways to extend them to a complete interpretation. This focusing of activity is brought about by inhibiting tasks looking for replacements for any of the focus phrases, unless the potential replacement promises to lead to a significant improvement in value for the final interpretation. Tasks conflicting with the focus of activity have their priority temporarily lowered so that the parser is biased toward building up a complete interpretation using phrases in focus rather than exploring competing interpretations that would not use focus phrases. If the focus is wrong, then the attempts to extend it to a complete interpretation will be unsuccessful. Eventually a task that conflicts with the focus will become the highest priority operation for the parser to perform in spite of the bias against it. As a result, the focus set will be modified so that it is consistent with the new task, and the parser will then concentrate on using the revised set of phrases.

In addition to calculating priorities of tasks on the basis of interpretation values and focus of activity, the parser must ensure that the information gained through the performance of the tasks is used effectively. This is done by structuring the parse net and the tasks that operate on it in a way that brings together related activities and coordinates them to eliminate duplication of effort. By avoiding duplication, the system reduces the ill effects of the false steps it will inevitably take. Work done on a false path is not necessarily wasted, since it may produce a phrase that can be used in some other way. For example, a phrase constructed as part of an unsuccessful search for one type of sentence may later appear in the final interpretation as part of a different kind of sentence. Also, false steps are not repeated, since the system only makes one attempt to build a particular type of phrase in a particular location in the utterance, regardless of how many larger phrases might include it. Mistakes are inevitable, but at least the system will not make the same mistake twice in one parse.

To summarize, the parser balances the desire to find the highest value interpretation of an utterance against the need to make a prompt response. In a step by step manner, phrases are created, evaluated, and combined. The choice of the next operation to carry out takes the form of assigning priorities to alternative tasks. Priorities reflect both the expected values of interpretations toward which the task would lead and the relation of the task to the current focus of activity. Finally, the entire process is organized so that information gained in performing a task is shared and recorded in such a way that it does not have to be rediscovered.

This sketch provides a rough outline of the parsing system. The remainder of this section gives a complete description, including overviews of the parse net data structure, the types of tasks and how they interact, the operations entailed in setting priorities, and the interfaces to other parts of the understanding system.[2]

B. The Parse Net

The parse net is the principal data structure built by the parser during its search for a complete interpretation of an utterance. Nodes in the parse net are either phrases or predictions for a certain category of phrase in a certain input

------

[2] In addition to the contributions made by the members of the SRI Speech Understanding Research Project, the design of the parsing system also benefited from critical comments by Jeff Barnett of System Development Corporation and Joyce Friedman of the University of Michigan.

location.[3]

1. Phrases

While an utterance is being parsed, the net contains many phrases for different categories and different parts of the input. The phrases can be either 'terminal' or 'nonterminal' and 'complete' or 'incomplete'. Terminal phrases correspond to words recognized in the input, and nonterminal phrases correspond to the results of applying composition rules to constituent phrases. An incomplete terminal phrase has only the lexical category and possible position specified but not the particular word. A complete terminal phrase can be constructed from an incomplete one by recognizing a word of the appropriate category. An incomplete nonterminal phrase has its rule and possible position specified but is missing one or more constituents. A complete nonterminal phrase can be constructed from an incomplete one by supplying complete phrases to fill the empty constituent positions. If all the constituents of a nonterminal phrase are missing, it is called an 'empty phrase'.

The left and right boundaries of complete phrases are given as times from the beginning of the utterance (in tens of milliseconds). With incomplete phrases, the system must deal with possible as well as actual positions. For instance, if an

------

[3] The design of the parse net was directly inspired by Kaplan's multiprocessing approach (Kaplan, 1973). It is also clearly related to the systems of Kay and Woods (see, for example, papers by them in Rustin, 1973).

incomplete nonterminal phrase is missing its leftmost constituent, then its actual left boundary is undetermined. Its possible left boundary can be specified either as a particular fixed time (in which case actual leftmost constituents must start at that time), or as a limiting time (meaning that leftmost constituents must not start before that time). Similarly, possible right boundaries can be either a particular time or a limiting time before which rightmost constituents must end.[4]

A large part of the parser's activity centers around making complete phrases out of incomplete ones. For terminal phrases, this requires identifying appropriate words in the input. For nonterminal phrases, it means constructing missing constituents. A complete phrase that results from an incomplete phrase A by supplying the missing word (if A is terminal) or the missing constituents (if A is nonterminal) is called a 'completion' of A.

------

[4] In the current system, there are actually four types of position specifications. In addition to the fixed point boundaries and the limit boundaries mentioned above, there are also 'range' boundaries and 'affix' boundaries. The range boundary is given by two points between which the actual phrase boundary must fall. The affix boundary is given in terms of a series of affixes and a point or range bound[illegible]. The actual phrase boundary must fall at a distance from [illegible] point or range leaving room for the affixes. Range and affix boundaries are not discussed in detail because they will be eliminated in the next version of the system.

2. Predictions

In addition to phrases, the parse net contains nodes called predictions. A prediction is initially created to reflect a missing constituent in some incomplete nonterminal phrase. From the rule pattern and time constraints of the phrase, it is possible to specify category and time constraints for the missing constituent, and these together serve to individuate a particular prediction. The category and position constraints of the prediction can be satisfied either by terminal phrases, if the predicted category has lexical entries, or by nonterminal phrases, if there are composition rules for the category. Just as there can be many ways to satisfy a prediction, there also can be many phrases waiting for the prediction to be satisfied, since the same prediction is shared by all phrases missing a constituent with the same category and time constraints. Thus predictions serve as intermediaries between sets of incomplete phrases, all missing a constituent of a particular category at a particular place in the input, and other sets of incomplete phrases that might supply the missing element.

3. Connections in the Parse Net

Most of the direct connections in the parse net are between predictions and phrases. There are no direct prediction-to-prediction connections, and the only direct phrase-to-phrase connections are the 'immediate constituent' links from nonterminal phrases to the complete phrases used to construct them. Complete phrases also have an 'instantiation' link from predictions that they satisfy. Figure III-1 shows the two kinds of links to a complete phrase: the immediate constituent link from a (complete or incomplete) nonterminal phrase and the instantiation link from a prediction. A complete phrase can be pointed to by many links of each kind--it can be a constituent of many phrases and an instantiation of many predictions. (The procedures that establish these and other connections in the parse net are discussed later in this section.)

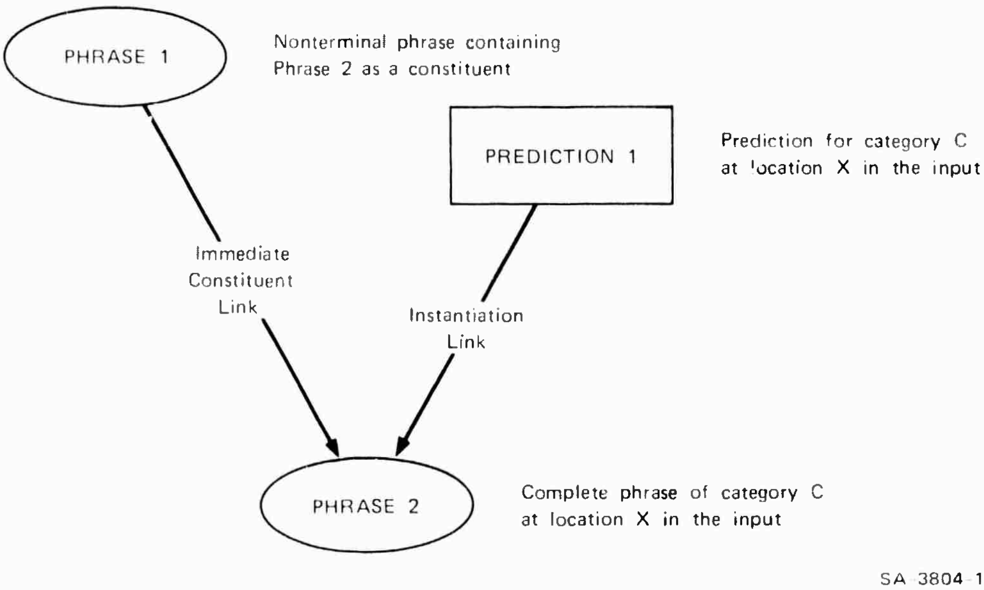


FIGURE III-1 LINKS TO A COMPLETE PHRASE

The direct connections between incomplete phrases and predictions are of two types (see Figure III-2). The bidirectional link between an incomplete nonterminal phrase and

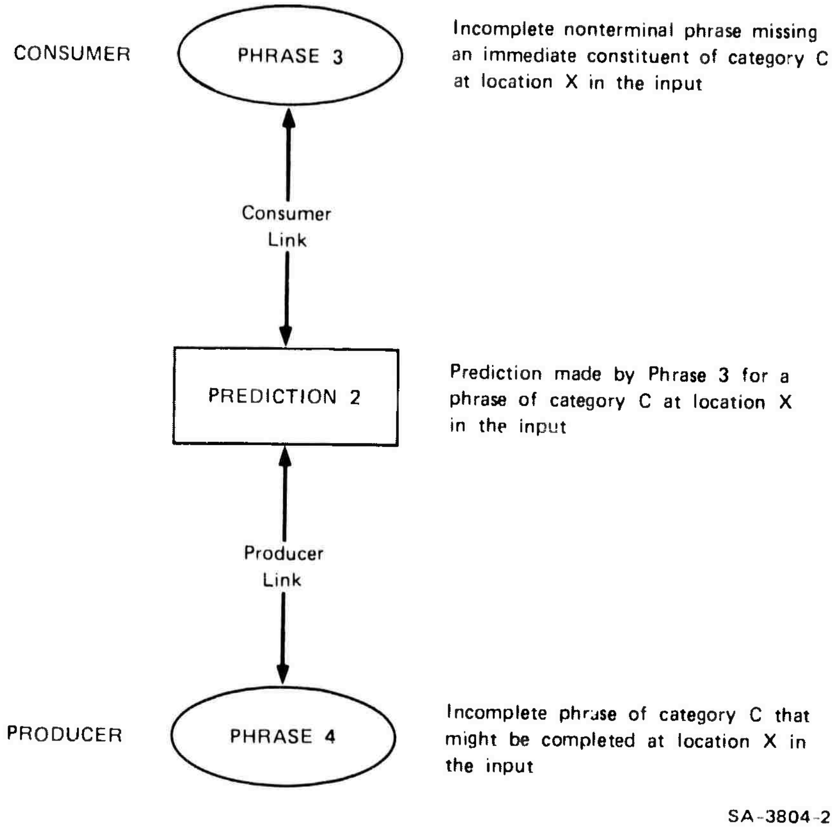


FIGURE III-2 CONSUMER AND PRODUCER LINKS one of its predictions is labeled a 'consumer' link, and the phrase is referred to as a 'consumer' for the prediction. This terminology reflects the fact that phrases produced according to the constraints of the prediction will be utilized as constituents of the consumer phrase. Similarly, the bidirectional link between a prediction and an incomplete phrase that satisfies the prediction's constraints is called a 'producer' link, and the phrase is referred to as a 'producer' for the prediction. This is because completions of the phrase produce constituents to be used by the consumers of the prediction. Note that the set of producers and the set of consumers are not disjoint classes of phrases; a phrase may be producer with respect to some predictions and at the same time a consumer for others. In general, a phrase may be a producer for any prediction whose constraints it satisfies and a consumer for any prediction it has made. Since a prediction may also have many consumers and many producers, the parse net is richly connected (and can even become cyclic as discussed below). A simple configuration is shown in Figure III-2. A consumer, Phrase 3, is joined by a consumer link to one of its predictions, Prediction 2, which in turn is joined by a producer link to one of its producers, Phrase 4.

Figure III-3 gives a specific example of this kind of configuration. Phrase 3 in this instance is an empty phrase spanning the input corresponding to rule S1 from the SRI language definition. Rule S1 produces phrases of category S from two

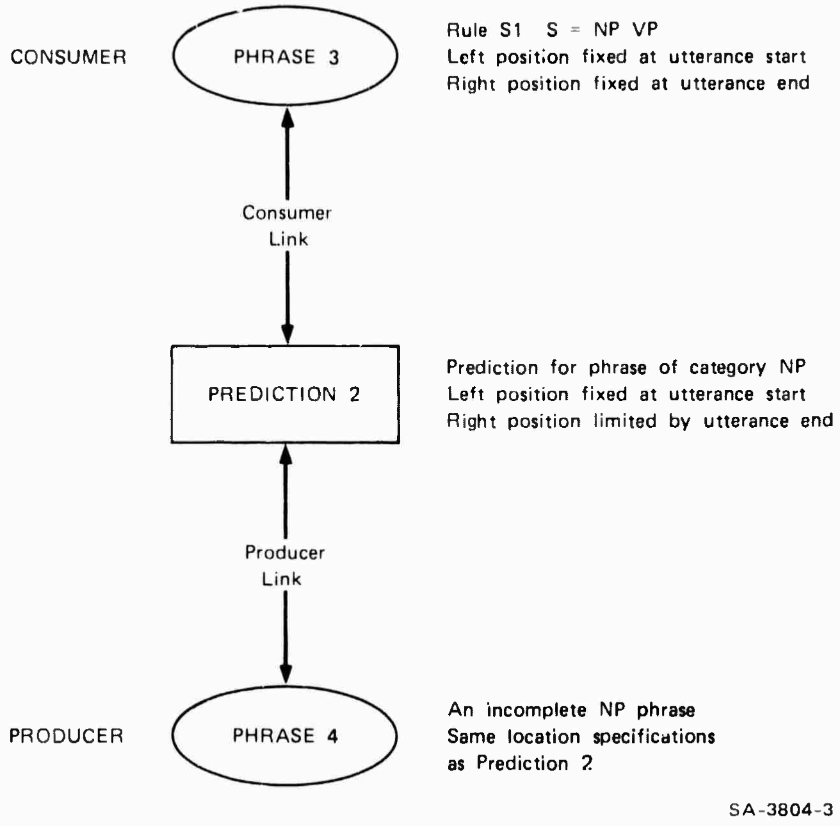


FIGURE III-3 A CONSUMER-PRODUCER CONFIGURATION constituents: a noun phrase (NP) followed by a verb phrase (VP). Phrase 3 is thus missing (in addition to the verb phrase) an NP starting at the beginning of the input, so there is a consumer link from Phrase 3 to a prediction for an initial NP. The prediction is linked to a producer phrase, Phrase 4, of category NP. Phrase 4 has its left boundary fixed at the start of the utterance and may be either a terminal phrase, in which case it can be completed by finding a word from category NP (such as "it"), or a nonterminal phrase, in which case it corresponds to some rule for constructing NPs (such as a rule combining a determiner and a nominal). In either case, a completion of Phrase 4 would become an instantiation of Prediction 2 and an immediate constituent of (a copy of) Phrase 3.

In addition to the direct connections in the net, some of the indirect connections are important for describing the operations of the parser. The (direct) consumers of a phrase are reached by following first a producer link from the phrase to a prediction and then a consumer link from the prediction to another phrase. Equivalently, the consumers of a phrase can be defined to be the consumers of the predictions for which the phrase is a producer. For example, in Figure III-3, Phrase 3 is a consumer for Phrase 4. Since the consumers are also phrases, they in general have consumers themselves unless they are root category phrases. This makes it possible to follow links from a phrase along a path of more and more indirect consumers until reaching an 'ultimate consumer'. Each such maximal consumer path from a phrase represents a potential context for the phrase. The collection of such paths plays an important part in determining the priority of completing the phrase.

The producer paths from a phrase are defined in a similar manner. The (direct) producers for a phrase are reached by following first a consumer link and then a producer link. In other words, they are the producers for the predictions for which the phrase is a consumer. The symmetry of the producer and consumer definitions means that producer paths are simply consumer paths viewed from the opposite direction. Thus, in Figure III-3, Phrase 4 is a producer for Phrase 3. Producer paths are discussed further in conjunction with the propagation of changes in the parse net.

4. Consumer-Producer Cycles

Because of recursion in the language, cycles can occur in consumer-producer paths so that a phrase can be a consumer-producer for itself.[5] This most often occurs with a phrase having a limit for either its left or right time specification rather than having both times fixed. For clarification, consider Figure III-3 in which Phrase 4 (like Prediction 2) has its left position fixed at the beginning of the utterance and its right position limited by the end of the utterance. Note that the limit is not as far left as it might be, since, to be of use in Phrase 3, the noun phrase must stop far

------
[5] This discussion can be skipped on first reading.

/6~

enough before the end of the utterance to leave room for a verb phrase. However, if limits were always set as tight as possible, in addition to Prediction 2 and its producer phrases, there would be duplicate sets of predictions and producers for initial NPs with slightly different right limits corresponding to different possible constituent strings making up the rest of the utterance. For example, there would be at least one for an AUXB NP sequence (for sentences like "What is it?"), a second for an AUXD NP VP sequence (as in "What do you know?"), and others for sequences beginning with a nominal (for cases in which the NP is used as a possesive determiner as in "The ship's speed is 30 knots."). All these cases would have slightly different limits but would cause essentially identical tests to be performed at the beginning of the utterance. Acoustic mapping, for instance, will be guided by the fixed left boundary and is not likely to be affected by a small change in the right limit. By discarding the small differences in right limits, all these searches can be merged into a single search for an initial noun phrase free to end anywhere within the utterance. The savings from merging the searches in this way more than compensate for the small loss in precision with respect to the right time limit.

As a result of restricting position limits to the boundaries of the utterance, left recursion in the language (i.e., the existence of rules allowing a phrase to begin with a subphrase of the same category) leads to consumer-producer loops for phrases with right position limits, and right recursion leads to loops for phrases with left position limits. Noun phrases exemplify both types of recursion: left recursion since an NP can begin with a determiner that can in turn be a possessive NP (thus the NP "the ship's speed" begins with another NP "the ship"), and right recursion since an NP can end with modifiers such as prepositional phrases or relative clauses that in turn can end in NPs (as in "the speed of the ship").

Although consumer-producer loops are most often associated with limit position specifications, they can occur even with both positions fixed if the language contains rules such that a phrase can be completely represented in the acoustic signal by a subphrase of the same category. Again noun phrases provide an example, since by ellipsis an NP can be reduced to a determiner alone, the determiner can be a possessive noun phrase, and the possessive suffix can be indiscernible in a spoken utterance if the NP is plural. This is illustrated by sentence 1.

(1) The Marx brothers' favorite joke is vulgar, but the Three Stooges' is obscene.

Consumer-producer loops are accounted for in the parser and are mentioned again in the following discussions.

C. Types of Tasks in a Parse

An initial characterization can now be given of the types of tasks performed by the parser. In general, the performance of a task entails modifying the parse net and scheduling new tasks to perform further modifications. The most frequent tasks in a typical parse are prediction tasks and word search tasks. When a prediction is created, it is first entered in the net and linked to its initial consumer. Empty nonterminal producer phrases for the prediction are created in a manner described below, and, if there are lexical entries for the predicted category, an empty terminal producer phrase is created along with an associated task to begin looking for words. If later the word search task finds a word in the input, then a complete terminal phrase is created, entered in the net as an instantiation of the prediction, and distributed to the consumers. When the word search task has exhausted all its possible candidate words, it prunes the empty terminal phrase from the net. If all the producers for a prediction are pruned, the consumers of the prediction are also pruned, since no more ways are available to provide their missing constituents.

The result of distributing a complete phrase X to a consumer C is a new phrase C' that is a copy of C with X added as an immediate constituent. The score of C' must be above a certain threshold or else C' is immediately discarded. Assuming the score is all right, the treatment of C' depends on whether it is complete or not. If C' is complete, then it too is distributed. If C' is incomplete, a task is scheduled to predict one of its missing constituents. Note that the original consumer phrase C is unaffected by the creation of C'; C remains in the net waiting for other phrases like X to be found, in which case C will be copied again to create another phrase like C' for the new constituent. For example, if C is Phrase 3 of Figure III-3, then X is an NP, C' is an incomplete S phrase, and the scheduled task will predict a VP with left position fixed equal to the right of X and right position fixed at the end of the utterance. Phrase 3 is left waiting for other NPs to be found at the start of the input.

As mentioned above, when a prediction is made, empty nonterminal producer phrases corresponding to each of the language definition rules for the predicted category are created along with a task for each new producer to make a subsequent prediction for one of its missing constituents. The prediction task begins by determining which of the missing constituents can be used as the basis of a prediction. Predictions are restricted to cases in which at least one of the left or right positions is a fixed boundary rather than a limit, so not all missing constituents will qualify.[6] The left position of a missing constituent is fixed if it is both the leftmost constituent of the phrase and the left boundary of the phrase is fixed, or the constituent immediately to its left is not missing. The right position is fixed similarly by either the right boundary of the phrase or the presence of a right neighbor. For example, if an empty phrase with two or more

------

[6] In the following discussion, fixed boundary time specifications include point, range, and affix boundaries (as defined in an earlier footnote). In other words, 'fixed' is used as the opposite of 'limit'.

constituents has both left and right positions fixed, then predictions are possible for both the leftmost and rightmost constituents. In the parse net as a whole, there are always at least two fixed phrase boundaries--namely, the beginning and end points of the utterance; boundaries also arise inside the utterance when words are identified. Thus, predictions are initially possible from both ends of the utterance, and, as words are recognized, internal predictions can be made as well.

Since it is often the case that more than one prediction is possible for an incomplete phrase, the problem arises whether to make all the predictions, only one, or some intermediate number. The argument for making all possible predictions is that any single prediction could get bogged down while one of the others might succeed and provide enough information to "rescue" the first. For example, consider an empty phrase . with both left and right positions fixed and two (missing) constituents named A and B. If P predicts both A and B, if A succeeds it will give added information and allow a more precise prediction for B, and the same will happen if B succeeds. As an illustration of how a new prediction based on more information can overcome problems that might stall the original, less precise prediction, consider predictions for B before and after a phrase for A has been found. Before, the prediction for B will have only the right boundary fixed, and thus attempts to construct a B phrase will be initially limited to a right-to-left search. After an A phrase is found, a new prediction for B can be made with both boundaries fixed so that the search for a B phrase can also proceed in a left-to-right manner adjacent to the A phrase. This is valuable because it can lead to acoustic tests with both word boundaries fixed, and such tests can sometimes succeed in finding an acceptable match where tests with only one boundary fixed would have marginal or unacceptable results. In addition to the added boundary information, the A phrase can also lead to syntactic and semantic expectations about the B phrase that can override inhibiting factors, such as low scores on acoustic matches, that could stall the original B prediction.

The argument for making only one prediction is that the instances in which a secondary prediction will successfully rescue a primary prediction will probably be infrequent, and the system would do better to concentrate on a single prediction rather than spreading its efforts over several. In essence, this argument says that the increase in reliability from multiple predictions is not worth the associated decrease in efficiency.

When faced with a choice between reliability at the cost of efficiency or efficiency at the cost of reliability, it is appropriate to look for another alternative. In this case, by exploiting the task structure and scheduling abilities of the system, it is possible to get the efficiency of the single prediction approach without giving up the extra reliability of multiple predictions. This is done in the following manner. Consider phrase P mentioned above that can lead to predictions for either A or B. After the first prediction is made (for A, say), the prediction task for P is rescheduled at a lower priority to make the second prediction (for B). If the first prediction is successful in finding an A phrase, a P' is created by adding the A phrase to a copy of P and used to predict a following B. If that prediction is also successful, the second prediction for P is unnecessary, and priorities will never fall to the point that the prediction task for P is reactivated. However, if the first prediction runs into difficulties and no other alternatives work out, priorities will fall and the second prediction will be made.

When multiple predictions are possible, the parser makes the leftmost prediction first, because acoustic mapping tends to be more effective starting from the beginning of a word than from the end. This causes the initial operation to proceed in a generally left-to-right manner. If all goes well, words are found in sequence from the left, and the input is understood without the use of secondary predictions. However, if progress stalls, causing priorities to drop, then lower-priority, alternative predictions are made entailing right-to-left movement.

D. Initiating and Terminating a Parse

During a parse, tasks are performed that modify the net and schedule new tasks. The series of tasks is started by an implicit prediction for a root category phrase spanning the input. Empty nonterminal phrases with associated prediction tasks are created for each root category rule, and an empty terminal phrase with an associated word search task is created for root category lexical entries. When the ability is developed to spot words in the input without waiting for them to be predicted, the initialization phase will also include entering 'spotted' words in the parse net and creating nonterminal phrases containing them as constituents. This will lead to predictions for possible constituents adjacent to the words. The parser then goes into a cycle of removing and performing the highest priority task on the queue.

There are several ways by which this cycle can be terminated. If there are no more tasks in the queue, the cycle must stop. The parser calls the response function declared in the language definition, telling it that all possible ways of interpreting the input have been considered. After the response function terminates, the parser returns control to the program that originally activated it.

The parser also calls the response function if it reaches any one of three limits, and stops if the limit is not relaxed. The limits are an upper bound on the number of tasks performed, a lower bound on the priority of tasks to be performed, and an upper bound on the amount of storage used by the parser. These limits are initialized by the program that invokes the parser and can be modified by the response function during the parse.

In addition to the calls mentioned above, the response function is also called whenever a complete interpretation is constructed. The parser does not automatically stop when it finds an interpretation; it is up to the response function to adjust the limits to control how much more is done to find others. By setting the lower limit on task priorities just below the value of the found interpretation, the response function can ensure that the search will stop before the parser begins looking for inferior alternatives. By setting the limit on the number of tasks performed just above the current number, it can affect how much longer the search will go on. The role of the response function is thus to collect interpretations, adjust the limits controlling the parser, and initiate a response based on whatever interpretations have been found when a limit is finally reached.

E. Phrase Values

The preceding sketch mentioned that each cycle of the parser starts by selecting the highest priority task. We now turn to the question of how task priorities are determined. In general, tasks are associated with an incomplete phrase and have the goal of contributing toward completing the phrase. For instance, word search tasks are to complete terminal phrases, and prediction tasks are to complete nonterminal phrases. The major consideration in setting the priority of such tasks is the 'value' of the associated phrase. A second consideration is the relation of the phrase to the current focus of activity. We first discuss the calculation of the value of a phrase and then turn to the question of focusing the parser by adjusting priorities.

1. Value of a Phrase

The 'value' of a phrase P is an estimate of the best value of an interpretation containing a completion of P. The value of an interpretation is derived from the score of the complete root category phrase forming the interpretation. (See the discussion of "Combining Factors into Composite Scores" in Section L, The Definition System.) The actual algorithm for deriving the interpretation value from the score is as follows. A constant K fixes the range of values as 0 to 100 times K.[7] A score is either an integer or a pair of integers <WEIGHT, TOTAL>. If the score is an integer, the value is K times the score. Otherwise, the value is K*TOTAL/WEIGHT.

The value of a phrase P is thus derived from an estimate of the root score of the best interpretation that can be built using P. To form such an estimate, we first need a representation of possible completions of P itself. If P is nonterminal, then it is clearly impractical to generate all possible completions of P to set the priority for completing P. Instead, P itself is used to represent the class of its possible completions.

------

[7] K is chosen according to the range of 'small' integers in the LISP implementation. See Teitelman (1974) for an explanation of small integers.

Recall from the discussion of the language definition system that rule attribute and factor statements are required to account for cases in which certain constituent attributes are 'UNDEFINED'. This lets the parser calculate attributes and factors for incomplete phrases by making all attributes of missing constituents equal the special constant UNDEFINED. The attributes of P can be assumed to reflect what is currently known about all completions of P based on the incomplete set of constituents. In some cases, the attribute may not depend on the missing constituents and will be the same in P as in all completions of P. In other cases, the attribute in P may reflect the range of possibilities in completions of P. For example, the semantic type of a noun phrase may be constrained, but not fully determined, when an adjective has been found but not the head noun. Finally, the attribute can be equal to UNDEFINED in P, if nothing can be determined until more constituents are fixed. In the same manner, the factors for P reflect estimates of the factors in successful completions of P, and thus P's score can be used as an estimate of the score for completions of P.

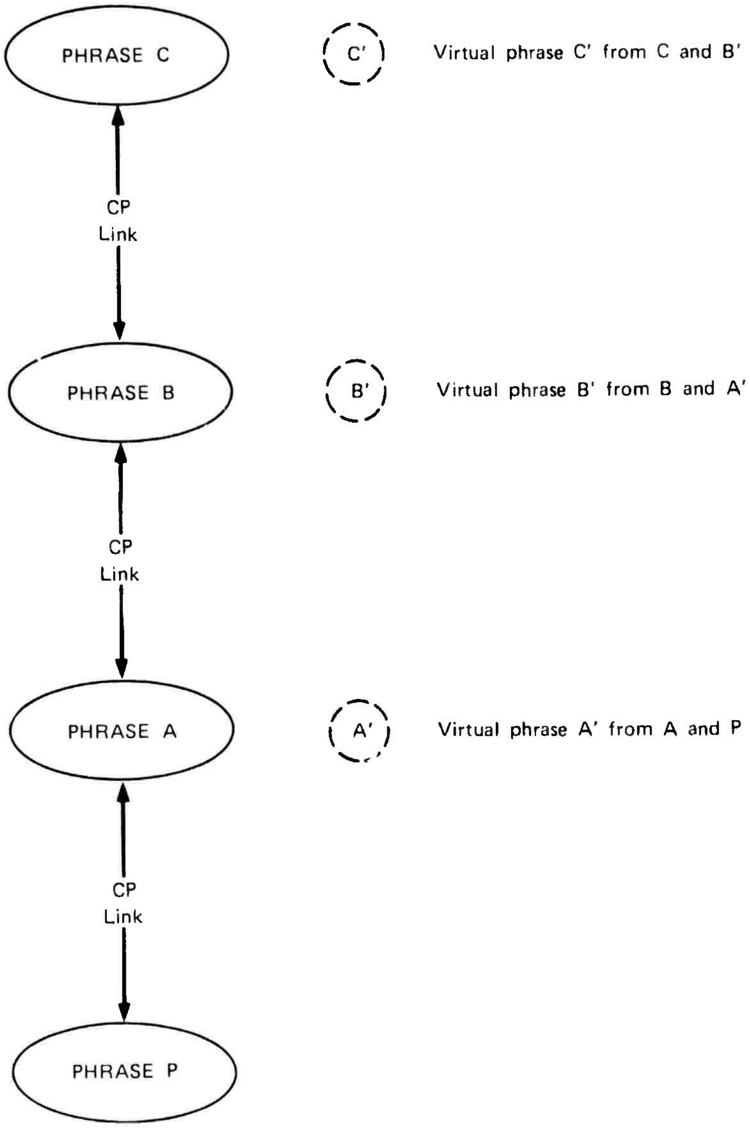
How do we get the value of P from its attributes and score? If P is itself a root phrase, P's score is the desired estimate. If P is not a root phrase, the parser must look at various ways of embedding P in root phrases. This can be done by exploring the consumer paths leading from P. Each path from P to a root category consumer reflects a way currently under consideration of constructing an interpretation using P. If an estimate can be made of the best value that would result from completing an interpretation based on a path (called the value of the path), the estimate for the best path can be used as a value for P.[8]

In calculating the value of a path, temporary structures called 'virtual phrases' are built based on the consumer phrases in the path. To make the discussion more concrete, let A-B-C be a consumer path for P (see Figure III-4). Phrase A is a direct consumer for P (i.e., a completion of P could fill an empty constituent position in A), B is a direct consumer for A, and C, for B. The virtual phrase A' is formed by placing P in the appropriate empty constituent position in A. In the same way that the attributes of P reflect possible completions of P, the attributes of A' reflect possible completions of A-plus-P. Similarly, the score of A' based on its factors can be used as an estimate of the score of A-plus-P completions.

In the same manner, B' is constructed using A' as a

------

[8] To allow for bottom-up parsing, consumer paths are not constrained to end in root phrases. Such incomplete paths only partially specify a way of constructing a complete interpretation, and therefore present a problem for calculating phrase values. In the current system, the expedient has been adopted of treating incomplete paths like complete ones that do reach root phrases. A second problem stems from consumer-producer cycles in the parse net. There does not seem to be any well-motivated theoretical limit on the number of times around a cycle a path should be allowed to go; a cycle thus represents an unlimited number of different potential paths. The current system (arbitrarily) resolves this problem by allowing a phrase to occur up to two times in a consumer path but no more. This corresponds to at most once around a loop.



CP link is an indirect link between a consumer and a producer via an intermediate prediction.

SA-3804-4

FIGURE III-4 A CONSUMER PATH

missing constituent in B, and C′ using B′ in C. The score for C′ gives the value of the path A-B-C with respect to P. The best value for any path from P gives an estimate of the value of the best interpretation using P, based on the current structure of the parse net--in other words, it gives the value of P.

In general, there can be many consumer paths from an incomplete phrase and many virtual phrases to be constructed along each path. Since building a virtual phrase requires evaluating attribute and factor statements and calculating the resulting score, the efficiency of the system can be improved by finding ways to reduce the number of virtual phrases constructed. The first method is to exploit the fact that the collection of paths forms a tree. For example, if in addition to the path A-B-C, there is another path A-B-D, then B′ has to be created only once for both paths (see Figure III-5). The attributes and score of B′ do not depend on whether it will be used with C to form C′ or with D to form D′. This changes the problem from finding the best consumer path from P to finding the best path in the consumer tree starting at P.

The cost of setting the value of P can be further reduced by avoiding an exhaustive search of this consumer tree. The first way of doing this is to ignore branches of the tree from virtual phrases whose score is below some threshold. The threshold is the same as is used to discard actual phrases; consequently, if the score of some virtual phrase, which reflects

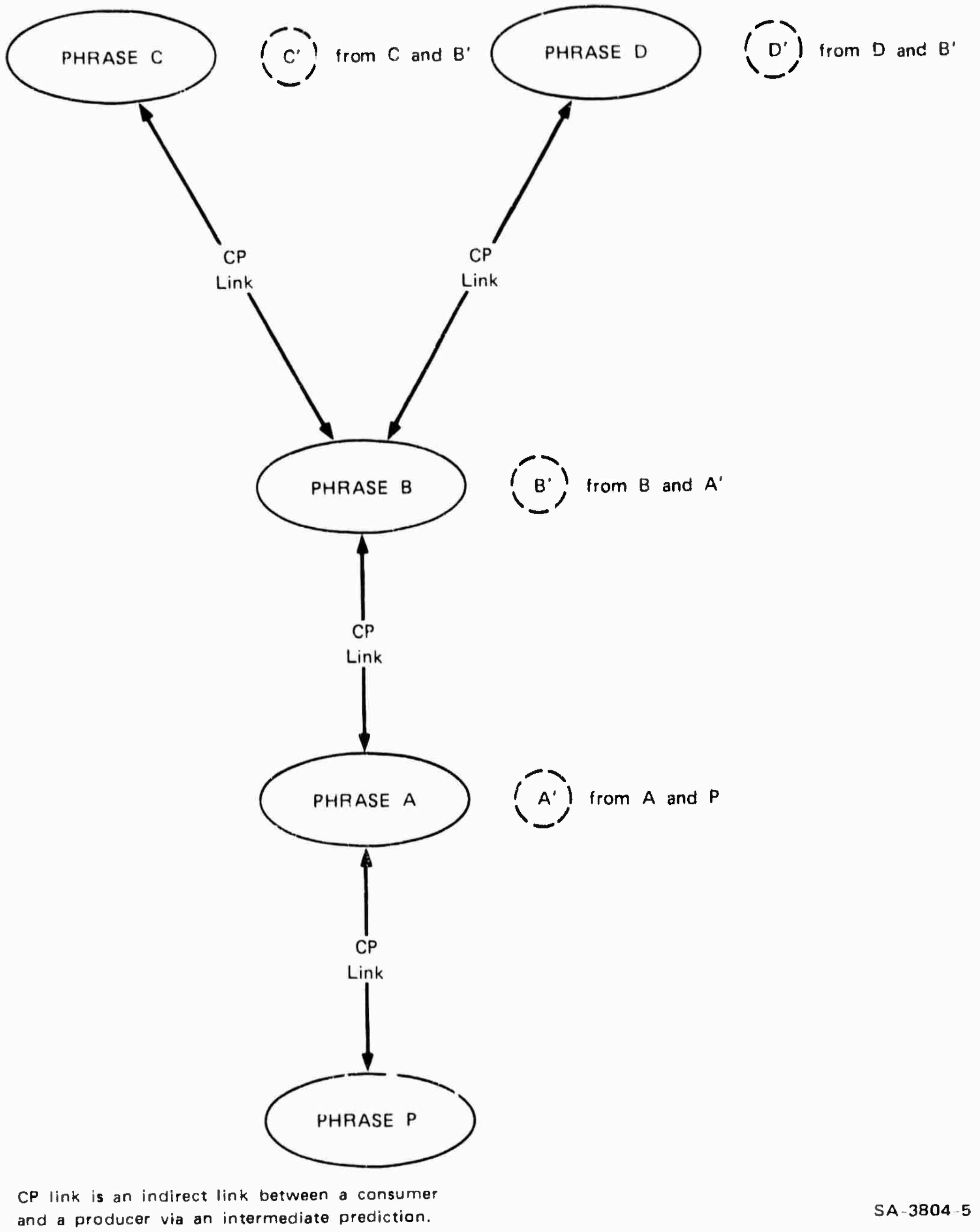


FIGURE III-5 A CONSUMER BRANCH

a particular way of using P, is below the threshold, it is likely that any actual phrase using P in that way would also have such a low score that it would be discarded. This means that since no interpretations would be formed using P that way, the value of that branch of the consumer tree can be safely set to zero.

The second way to reduce the amount of the consumer tree actually explored is to drop the requirement of finding the path that actually gives the best value and to perform a heuristic search for a path that is likely to give a result close to the best value. Notice that if the system keeps a running approximation of the value based on the portion of the consumer tree explored so far, exploring a new branch of the tree can only cause the approximation to go up since the goal is to find the best path. If it is possible to calculate for each branch a rough estimate, called the 'heuristic value' or 'HV', then the system can explore the branches with the highest HV first and skip branches with an HV lower than the approximation established to that point. If the HV is always a true upper bound on the branch's actual value, this search will find the best value. If the HV can be low by up to ten percent, say, the search may produce a value as much as ten percent suboptimal. However, by giving up optimality, the amount of search typically required can be significantly reduced, since lower HVs lead to more skipped branches. For this reason, the system uses an algorithm for calculating heuristic values that is not guaranteed to yield a true upper bound but that should rarely fall far below.

In calculating an HV for a consumer branch that starts with a particular consumer C, the parser takes advantage of the fact that the branch has already been searched in the process of determining the value of C when C was created. The HV algorithm merges the previously calculated value of C with the score of the phrase X, which is to be added to C to form a new virtual phrase. The score of X has the form <WEIGHT(X), TOTAL(X)>. Saved with C is its value and the weight part of the score at the end of the consumer path that was used to derive the value. This makes it possible to form an estimate <WEIGHT(path) + WEIGHT(X), alpha + TOTAL(X)> of the score that would result from exploring the branch above C with respect to X (alpha is essentially TOTAL(path)[9]). This score is then converted to a value in the manner explained above and used as the HV for the branch.

Notice that all phrases with the same score as X will get the same HV with respect to the consumer branch starting at C. This reflects the fact that the heuristic value is independent of the detailed requirements of the consumers and the attributes of X. If X satisfies the consumer requirements, the HV will be a reasonably good approximation of the actual value; if X violates the requirements, the HV may be much too high. For this reason, the heuristic value cannot in general replace a search of the

------
[9] Alpha is computed according to the formula Value(C)*WEIGHT(path)/K. Since by the definition of value, Value(C) is K*TOTAL(path)/WEIGHT(path), alpha differs from TOTAL(path) only because of roundoff errors introduced by integer arithmetic.

consumer tree as a means of establiching the actual value, but it can be effectively used to limit the search.

In summary, the procedure for setting the value of a phrase P has the following form. The value of the consumer branch for P with the highest heuristic value is calculated. Then the consumer branch with the next highest HV is selected. If its HV is not greater than the approximation already determined, the process terminates. Otherwise this consumer branch is evaluated, and the result is used to update the approximation. This cycle continues until all the consumer branches are evaluated or rejected because of heuristic values lower than the approximation. A similar algorithm is used to explore the branches of the consumer tree from a virtual phrase C'. The consumer branches for C' are searched in order of HV as long as the HV is greater than the current approximation for P's value.

In the jargon of heuristic programming, this is a depth-first search with forward pruning and generation of successors in order of their estimated worth (see Nilsson, 1971). It should be possible to reduce the amount of the consumer tree explored even further by changing to a best-first search method in which paths are suspended whenever alternative paths exist with a higher heuristic value. Whether the savings from reduced search would compensate for the increased overhead of the more complex search method remains to be seen. This question will be considered further if measurements suggest that the cost of determining phrase values is significant.

2. Value of a Terminal Phrase

The preceding discussion has dealt with setting the value of incomplete nonterminal phrases. The process is essentially the same for terminal phrases, except that it is performed for particular words that might complete the phrase rather than for the incomplete phrase itself. Initially, the alternative words are all assigned a value equal to the value of the nonterminal phrase making the prediction. Since the alternatives are ordered, the words tried first are those most likely to be really present if they are accepted by the acoustic matching procedures. Typically, this implies trying long words before short ones. When the word search task is performed, the first alternative is removed from the list and given to the lexical factor function (see the section on the internal representation of the language definition). The factor function, which includes calls on acoustic mapping routines, either accepts the word, in which case a complete terminal phrase is created, rejects the word, in which case the alternative is deleted, or requests to reschedule further processing on the word. In the last case the value of the alternative is calculated in the manner described above and the word is returned to the candidate list. The cycle of accepting, rejecting, or rescheduling the highest value alternative continues until all the alternatives have been eliminated (either by acceptance or rejection) or the highest value is lower than the value of the first alternative tried. If the value has dropped, the phrase value is reset and the task is rescheduled at a lower priority.

F. Focus of Activity

The priority of both word search and prediction tasks is initially set equal to the value of the incomplete phrase with which the task is associated. In both cases, the priority is lowered if the associated phrase conflicts with the current focus of activity for the parser. This section discusses why this extra step has been introduced in setting priorities, how focus is established and revised as the parse progresses, and how conflicts with focus are detected and 'punished'.

The value of a phrase reflects its score and its consumer context but not its competition. If an incomplete phrase P has a high value, other phrases similar to P are also likely to have high values. If values alone determined priorities, then even after successfully filling the empty constituent positions of P to form a complete phrase P', the parser would tend to continue looking for slight variations on P' in the same area of the input, rather than moving on to look for ways to use P' to construct a complete interpretation. The focus mechanism provides a way for phrases like P' to inhibit the search for other phrases that would necessarily replace them in a complete interpretation. The inhibition is brought about by lowering the priority of tasks that would lead to the creation of such competitor phrases. Inhibiting competition has the effect of focusing the activity of the system on finding ways to use the phrase. This technique balances the goal of finding the highest value interpretation against the goal of making a prompt response.

1. Placing a Constituent in Focus

At any given time during a parse, the current focus is represented by a possibly empty set of nonoverlapping complete phrases. As the parse progresses, the focus is automatically established and adjusted by revising the contents of the focus set. In the organization of the parser, setting and modifying focus are tied to making predictions. Before making a prediction, the parser checks whether the phrase P making the prediction conflicts with the focus. If there is a conflict with some focus phrase F, the conflict is either resolved in favor of F, in which case the prediction task is rescheduled at a lower priority, or in favor of the prediction task, in which case F is removed from focus. Thus phrases can be removed from focus if they conflict with a task that becomes highest priority in spite of being inhibited by its conflict with focus. Assuming that the task for phrase P either did not conflict with focus or has forced the removal of any conflicting phrases, the parser's next step is to make a prediction for P. To bias further work in favor of P, the parser then proposes constituents of P for addition to the focus set.

Constituents are inspected before being placed in focus, since inclusion represents a commitment by the system to try to use the constituent in the final interpretation. Both the score of the constituent and its likelihood of false acoustic acceptance are considered, and only phrases meeting certain criteria are allowed into the focus set. Even phrases that are added have their inhibitory strength adjusted according to the system's confidence that they are correct.

2. Factors Controlling Focus Strength

The inhibitory strength of a focus phrase is an integer indicating the percentage by which the priority of a conflicting task is reduced. The strength determines both how much the phrase inhibits conflicting tasks and, through that, how resistant it is to being removed from focus. If an infallible oracle gave assurance that a certain phrase was correct, the phrase could be put in focus with insurmountable strength so that it would correctly eliminate all attempts to replace it. Lacking reliable oracles, the system must limit the strength of focus phrases to reflect the uncertainty associated with them.

Four factors control focus strength: the score of the phrase, the likelihood that the phrase has been incorrectly accepted by the acoustic routines, the value of the phrase putting it in focus, and the presence of immediately adjacent phrases in focus. The first two factors reflect the system's confidence in the phrase in isolation. The third factor shows how well the phrase fits into the total context of consumers. The final factor is based on the observation that a phrase is less likely to be incorrect if good phrases can be constructed on either side of it. In the current implementation, if all the factors are favorable, the focus strength is set to produce about a ten percent decrease in priority. This amount is large enough to have a significant impact on what tasks are performed but small enough to allow the system to recover from occasionally putting an incorrect phrase in focus.[10]

3. Changing Focus Strength

The strength of a focus phrase F can change as the parse

------

[10] From limited experimentation, it appears that a major cause of incorrect focus phrases in the current system is erroneous closure, in which a proper subpart of a correct phrase is mistakenly taken to be the complete phrase. For instance, this can happen if the language includes rule patterns such as (1) A = B and (2) A = B C. An incorrect use of the first rule where the second should actually apply would be an instance of erroneous closure. Rules such as 1 and 2 are common (as seen in the current SRI language definition), which is why erroneous closure has a large potential for producing incorrect phrases. A possible approach to dealing with this (that we intend to study) is to use one symbol lookahead to adjust the priority of applying rules such as 1 that can produce incorrect closures. Such lookahead depends crucially on acoustic capabilities such as lexical subsetting (discussed in Section H, Interfaces). Conjectures about human parsing strategies suggest that one symbol lookahead should be helpful in parsing a language such as English (see, for example, Bever, 1970, Grosu, 1972, and Kimball, 1973)--cases in which more lookahead is necessary tend to be either ruled out by the structure of English or difficult for people to comprehend.

progresses. The strength increases when a more valuable phrase puts F in focus or when F becomes bounded on both sides by other focus phrases. The strength of F decreases if a neighbor of F is removed from focus causing F to be no longer bounded. Finally, removal from focus can be viewed as an extreme case in which the strength vanishes. The two kinds of strength changes, increases and decreases, are treated differently. Decrease in strength causes conflicting tasks to be immediately raised in priority. (Since the focus phrase carries with it a list of conflicting phrases, each of which in turn has a list of associated tasks, the relevant tasks needing priority increases are easily located.)

On the other hand, an increase in strength does not lead to an immediate priority drop for conflicting tasks. Instead the system waits until the task is the highest priority task before lowering its priority. For instance, when a prediction task is activated, one of the first operations is to check if it was already in conflict with some phrase and if that phrase has increased in strength since the conflict was recorded. If the strength has increased, the task is rescheduled at a lower priority. Otherwise, the focus phrase is removed from focus, and the prediction task continues. By delaying priority decreases resulting from increases in focus strength, the system can avoid unnecessary rescheduling of tasks that are already of such low priority that they are unlikely to be activated.

4. Focus Conflicts: Word Search Tasks

Having discussed how focus is established as part of prediction tasks and how focus strength is determined, the next topic is how focus conflicts are detected and dealt with. There are two cases to consider: conflicts affecting word search tasks and conflicts affecting prediction tasks. The type of conflict considered for a word search task is called an area conflict. The time specifications of the incomplete terminal phrase associated with the word task determine an area of the input that any word completing the phrase would have to include. An area conflict simply means that some focus phrase already occupies at least part of that area of the input. If there is no such conflict, the word search proceeds. If there is a conflict with some focus phrase F, the word search is rescheduled at a priority reduced according to the inhibitory strength of F. If the word task becomes top priority in spite of this conflict, it is marked 'immune' to F. The area conflict check is repeated, but this time ignoring F (and any other focus phrase for which this task is immune). If there is a conflict with a focus phrase weaker than F, the word task also becomes immune to it. In case of a conflict with a focus phrase stronger than F, the word task must be rescheduled again at a still lower priority. Otherwise, the task is performed in the manner sketched earlier.

5. Focus Conflicts: Prediction Tasks

The procedure to check for focus conflict as part of a prediction task begins with a test for an area conflict based on the area of the input that any phrase instantiating the prediction would have to include. This has the effect of encouraging predictions in areas where the parser has not yet found a good phrase. If there is an area conflict with a focus phrase F, the task is rescheduled at a slightly lower priority (two percent lower in the current implementation). The amount of inhibition is independent of the strength of F in this case, because the area check is intended to provide only a slight push into new areas; other focus tests follow the area check and can produce inhibition proportional to the strength of the focus phrase. Also, the area check does not consider the possibility that the focus phrase may actually be compatible with the prediction. In other words, it may be possible to satisfy the prediction without removing F from focus. This is a second reason for keeping the inhibitory effect of area conflicts on prediction tasks small and independent of the strength of the conflicting phrase.

If there is an area conflict, the task is rescheduled. When it becomes highest priority again and is reactivated, it goes on to the next step of the prediction procedure as if no conflict had existed. The second stage of focus checks entails testing the immediate constituents of the phrase making the prediction. A constituent phrase C conflicts with a focus phrase F if C overlaps

F but is neither identical to F nor contains F as a subphrase. If there is no overlap or F is equal to or contained in C, clearly there is no necessary conflict between having both C and F in the final interpretation. If there are no constituent conflicts, the focus tests progress to the third and final stage, tests for conflicts between focus and the constraints on missing constituents.

For each missing constituent in the predicting phrase, the category and position specifications are tested against the focus. The details of the test vary according to the position specifications; there are separate tests for each of the four combinations of left and right, fixed or limit. Since the cases are similar in general structure, we will sketch only one as an example (see Figure III-6). In a parse that is progressing in a generally left to right manner, the left-fixed and right-limit case is very common. The left position comes from the start of the utterance or the right boundary of the preceding phrase, and the right limit is typically the end of the utterance. If no focus phrase starts at the left boundary of the prediction, there is a conflict only if some focus item starts before the boundary and ends beyond it (Figure III-6a). If a focus phrase F has a left boundary equal to the left position of the prediction, there are three subcases to consider depending on the relation of the right boundary of F to the right limit of the prediction. If F extends beyond the limit, there is no way F could occur as part of any phrase satisfying the prediction, and a conflict exists

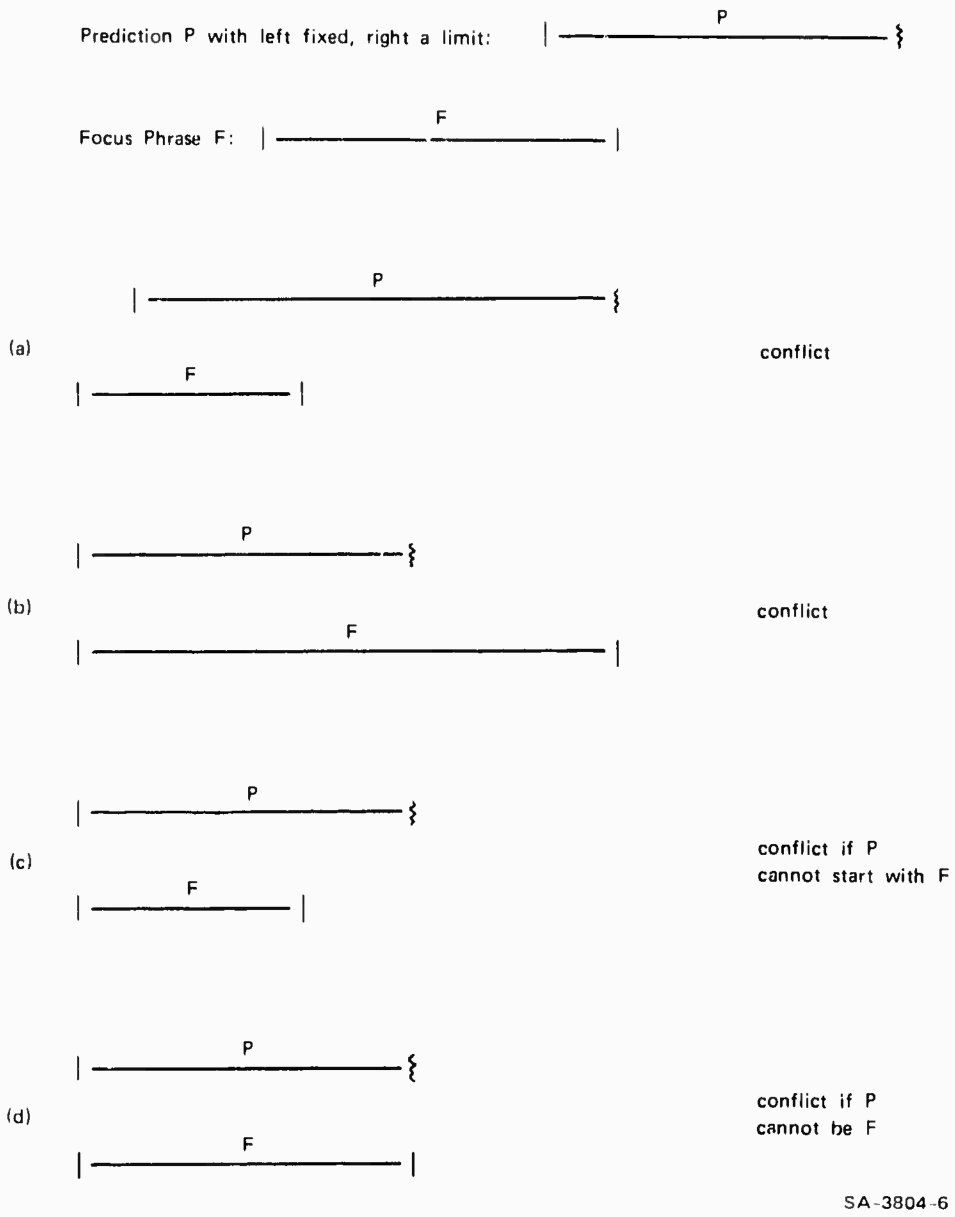


FIGURE III-6 FOCUS CONFLICT CASES

(Figure III-6b). If F ends to the left of the limit, a conflict exists if there is no chance that F could occur as a leftmost constituent of a phrase of the predicted category (Figure III-6c). Similarly, if F ends exactly at the right limit, there is a conflict if F cannot be completely dominated by the predicted category (Figure III-6d).

During the compilation of the language definition, special focus tables are constructed to facilitate tests such as the last two. The tests for focus conflicts make use of four precalculated lists for each category C: the categories that can occur as leftmost constituents of C, as rightmost constituents, as a constituent somewhere within a C, and as a complete C. Consequently, in case the focus phrase F starts at the left boundary of the prediction and ends before the right limit of the prediction, there is a focus conflict if the category of F does not occur in the list of categories that can be leftmost constituents of the prediction category, and similarly if F ends at the right limit.

In summary, with a fixed left boundary and a limit for a right position, a prediction conflicts if a focus phrase overlaps the fixed boundary or starts at the boundary but is inappropriate for the predicted category and right limit. The tests for the other combinations of left and right, fixed or limit, are carried out similarly.

6. Resolution of Focus Conflicts

If the checks made during the prediction task find a conflict with some focus phrase F, the conflict must be resolved before the prediction is made. The resolution depends on the relative strengths of F and the task. The strength of a focus phrase has already been discussed, and the strength of the prediction task is simply the maximum strength of any focus phrase it has conflicted with but overcome. Thus, if the task has already overcome the inhibition of a phrase as strong as F, the conflict is resolved in favor of the task and F is removed from focus. Otherwise, the conflict resolution favors F and the task is rescheduled at a lower priority. If the prediction task later becomes top priority in spite of this conflict, its strength will increase to equal that of F, F will be removed from focus, and the focus tests will be repeated except for the area conflict test. The strengthened prediction task either will conflict with a still stronger phrase in the revised focus set or will go on to make a prediction and add constituent phrases to focus.

G. Propagation of Consumer Changes

Change in focus strength is one possible cause of priority change as mentioned above, but it is not the only cause. In addition to depending on the current focus, task priority also depends on the value of the phrase associated with the task;

132

therefore, value changes need to produce priority changes. A phrase can change in value whenever there is a change in its consumer tree. The addition of a new consumer may cause the value to rise, and deletion of a consumer may cause it to fall. Additions occur when the same prediction is made by more than one phrase. Deletions occur when phrases are pruned. Two further types of changes are possible: a path that previously ended in a nonroot category phrase can be extended, or a consumer phrase can change in score. Currently, score changes occur only by human intervention; but if future systems are able to reconsider scores, then the mechanisms for propagating the changes will be available. The four types of changes can occur either for direct consumers or for consumers separated from the phrase by a path of intermediate consumers. This leads to eight types of consumer change to be dealt with. Because the similarities among the cases are more interesting than the differences, the discussion will be limited to additions. These are the most frequent, since making predictions is such a common operation, and they illustrate the important issues related to propagating consumer changes.

The addition of a new consumer for a prediction can potentially change the value of any producer for the prediction--direct or indirect. Consequently, the arrival of a new consumer is news that must potentially be propagated throughout the entire producer tree--potentially, rather than actually, because the propagation does not have to occur all at once and does not have to be completed before an interpretation is found.

Ideally, the change would be propagated only when and where it would make a difference in the choice of the task to be performed next. To approximate this condition, the system propagates the change step by step from a phrase to its direct producers. The task priority for propagating a consumer addition from P is set to the value that P has with respect to the addition--in other words, the value that P would be assigned by considering only the paths in P's consumer tree that begin with the path from P to the new consumer. If the addition is propagated to a phrase that conflicts with it in some way, the value with respect to the addition will be low, and the propagation will temporarily halt in that part of the producer tree. The overall effect is to avoid effort spent in making low value changes. This is another example of using the ability to schedule a task rather than performing it immediately in the hope that the procrastination will be rewarded by unnecessary tasks remaining undone when the parse ends.

H. Interfaces

The design of the language definition system simplifies the problem of interfacing the parser to the other parts of the understanding system. Most parts contribute to the parse through attribute and factor statements and have no special interactions with the parser. This reflects the fact that the factor and attribute statements provide a general mechanism for a variety of sources to contribute to assigning values to phrases, values that provide a large portion of the information the parser needs in order to set priorities.

1 Factors and Attributes

The acoustic component, for example, gives its information to the parser mainly through attributes and factors (but not exclusively--the other methods are discussed below). There are factors reflecting acoustic mapping both for individual words in terminal phrases and for sequences of words and affixes in nonterminal phrases. The word (or sequence) and the position specifications proposed by the parser are matched by the acoustic routines against the input to produce both a score indicating degree of match and actual positions if a match existed. The score is used as the basis of a factor, and the positions are used to set the left and right boundary attributes of the phrase. The mapping allows for the possible contexts the phrase might occur in, and is therefore especially lenient regarding the areas near the phrase start and end. This procedure makes it possible to share the results of the mapping among different consumers, but it also has the negative effect of causing small words, which are nothing but a start and an end, to be frequently accepted when they are not really there. The word sequence mapping in nonterminal phrases provides a chance to catch such errors by considering more context. When the small word is remapped in conjunction with proposed neighbors, the matching process can be less .giving. Coarticulation effects can be taken into account in checking word ends, where before lack of context made this impossible.

Other parts of the system also contribute information to the parser through factors and attributes. For example, semantic information currently comes to the parser solely through factors that rule out uninterpretable phrases, and syntactic information comes largely via factors (in addition to the syntactic information contained in the composition rule patterns). As long as the general attribute and factor mechanism suffices, the parser is unaffected by changes in knowledge sources or even by the addition of new sources. However, particular sources of information may have more specific advice to contribute than can be efficiently given through the means of factors and attributes. This is already the case with acoustic processing, as discussed below, and it may become the case for other parts of the system as well.

2. Dealing with Gaps and Overlaps

In addition to factors and attributes determined by the acoustic processes, the parser needs acoustic knowledge to deal with gaps and overlaps of adjacent phrases. This is necessary because the mapping routines cannot be expected to determine precisely accurate word boundaries. Some misalignment of boundaries is inevitable, but the amount of gap or overlap to tolerate and the amount to reject as excessive depend on details of the acoustic component. The parser needs information about allowable gaps and overlaps so that it can avoid constructing nonterminal phrases with constituents so far out of alignment that the phrase mapping will inevitably fail.

In the current implementation, the parser will not consider a phrase A as an immediate left neighbor of another phrase B if there is a gap or overlap of the right of A and the left of B greater than 0.6 seconds, or if B is not 'rightward' of A. B is rightward of A if either the start of B is to the right of the start of A, or alternatively the end of B is to the right of the end of A.

These constraints are so loose that any combination violating them can be safely discarded. Unfortunately, the looseness also means that many wrong combinations will not be filtered out. To deal with these, the parser makes a quantitative measure of the fit between a pair that passes the first test. If the fit is good enough, which will be the case if the gap or overlap is less than 0.2 seconds, the parser goes ahead constructing the phrase and lets the word sequence mapping confirm or reject the pair. However, if the fit is poor, but not so bad as to be clearly impossible, the parser reschedules the construction of the phrase at a priority reduced according to the degree of mismatch.

3. False Acceptance Estimates

The parser also makes use of acoustic information in the form of estimates, based on knowledge of the mapping routines, of the likelihood of incorrect acceptance for the various words in the lexicon. Currently this information is provided simply as a number from 0 to 100 for each word in the lexicon, giving a subjective probability that the word may not actually be present in the input even if it matches acoustically with a nonzero score. Small words like "a" and "of" get estimates near 100; larger words get estimates closer to 0.

This information is used in four ways. The first is to order word lists so that, other things being equal, the system proposes first the words that are the most likely to be really there if they are accepted by the mapping procedures. The second use is in converting mapper scores to factor scores--the range of factor scores is progressively narrowed around low-to-moderate values as the likelihood of incorrect acceptance increases. The third use is in setting focus and determining focus strength (see Section F, Focus of Activity).

The final use of the false-acceptance estimates is related to the distribution of newly created complete phrases. If there are no consumers or if none of the existing consumers can successfully combine with the new phrase, the parser has the option of creating consumers for it--a process referred to as 'bottom driving'. There may be no consumers if the phrase was found while searching for something else. For example, before mapping a word with both left and right times fixed, the parser tries mapping it with the right time a limit. If this attempt fails, then the original mapping with both times fixed is given a lower priority. However, the left-fixed and right-limit mapping may succeed but produce a complete terminal phrase with a different right boundary than the one originally predicted. In this case, the phrase may not have any consumers.

Another case that can lead to bottom driving in the current system concerns a phrase that instantiates some prediction, and thus typically will have some consumers but cannot be successfully combined with any of the consumers. This can happen if the phrase meets the category and time requirements of the prediction but has attributes that cause consumer factors to reject it. In either case, consumers can be formed corresponding to the composition rules with patterns including the category of the new phrase. This is worth doing if the consumer-less phrase has a good score and is not likely to be an unfortunate side effect of the mapper's difficulties in saying "no". But if the false-yes estimates suggest that the phrase would be difficult for the mapper to reject, the phrase is simply left in the parse net 'unconsumed'.

4. Lexical Subsetting

The use of false-acceptance estimates and the procedures for dealing with gaps and overlaps are instances of the parser incorporating general acoustic information relevant to processing any utterance. Both contrast in this respect with the next mechanisms to be discussed, lexical subsetting and word spotting, which provide specific information about the particular utterance being processed. Lexical subsetting reduces the number of words the parser needs to consider at a given place in the utterance, and word spotting gives the parser words from the input to use as additional starting points. (A lexical subsetter has been developed at SDC and will soon be combined with the parser for testing in a complete system. Work leading to a word spotter is in progress.) The subsetter will be activated as part of word search tasks. Before testing for words starting at position P, for example, the parser will call the subsetter to determine which words out of the lexicon might start at P. This subset of the vocabulary will be formed by considering robust acoustic features in the input signal directly to the right of P. The lexicon will have been preprocessed, so that given a particular subsetting feature, the words that might start in an area of the input with that feature will be directly available.

The result of the lexical subsetting will typically be used to eliminate a large percentage of candidate words. When a word is predicted at a particular location, the lexical subset at that location is searched and the candidate word rejected if it is not a member of the subset. However, in case the subset contains only a small number of words (perhaps four or so, excluding those with a high likelihood of false acceptance), the parser will not wait for the words to be predicted but will immediately propose them for mapping against the input signal. In this respect, subsetting foreshadows word spotting, the final interface between the parser and the acoustics.

5. Word Spotting

The next step beyond the use of reliable acoustic features for subsetting the lexicon is to use them in combinations to guide word recognition without waiting for the parser to propose particular words in particular parts of the input. Word spotting in this manner would be another way for the parser to get started in addition to predicting a root category phrase. Words located during an initial pass over the input would form additional starting points for the parser from which it could construct larger phrases in a more data-driven, bottom-up manner.

As experience is gained with this type of facility and as the acoustic capabilities increase, we expect the parser to evolve more refined methods for dealing with phrases that are found without being predicted. A nontrivial 'serendipity' problem is associated with relating unpredicted phrases to the rest of the parse net and coordinating the attempts to use them with other activities. The system currently reacts to phrases with no consumers by either creating all possible direct consumers or by creating none. A better solution might be to try to create selectively one or more chains of consumers to link the phrase to the existing parse net. How this should be done or whether a more radical change to the parser is needed are unresolved questions. This is another case in which the components of the understanding system must evolve together. Large changes to acoustic processing abilities will certainly lead to corresponding revisions in the parser.

I. Conclusions

The most distinctive features of the parsing system are the use of focus and phrase values in setting task priorities, the use of the parse net as a mechanism for coordinating activities and sharing results, and the use of automatically compiled, special purpose representations of the language definition. Of the three, we feel that the work on priority setting is both the most original and the most important. The use of a human-oriented, external representation and the precalculation of different, computer-oriented, internal representations has been very effectively used elsewhere, most notably in translator writing systems for programming languages.[11] The parse net structure follows the work of Kay, Kaplan, and Woods (see, for instance, papers by each of them in Rustin, 1973) and was directly inspired by Kaplan's multiprocessing approach (Kaplan, 1973). The method

------

[11] See Feldman and Gries (1968) for a discussion of translator writing systems. Recent work by Sager and her colleagues is an example of the value of special purpose 'metalanguages' for dealing with English; see, for example, Sager (1973), Grishman (1973), and Hobbs (1974).

of setting priorities according to a hierarchy of factors, scores, and values, with adjustments made according to a shifting focus of activity is an original contribution of this work. We feel it is a significant step toward dealing with one of the most difficult and important problems facing an understanding system: what to do next?[12]

The parser is currently written in SDC INFIX LISP and runs both on the IBM 370 with the SDC LISP system and (via a translator) on the DEC PDP 10 in INTERLISP. Preliminary tests have been made with real and simulated speech input and have lead to a variety of changes, mostly minor. The tests with actual speech, which must be run with the SDC LISP system, have been hampered by LISP storage limitations. These problems will be resolved by conversion to a new programming system, CRISP (Barnett and Pintar, 1974), which will be compatible with SDC INFIX LISP but will provide both more storage and efficient data structure facilities.

The evolution of the parser will be guided by internal pressures resulting from tests and measurements of its performance and by external pressures resulting from changes in other components of the understanding system. Until more extensive

------

[12] The parser design was influenced by a great deal of other research on the problem of parsing spoken input. Of particular relevance were the projects reported in Ritea (1974), Lesser, et al. (1974), Woods (1974), and Miller (1974).

tests are made, it is difficult to predict what direction the changes will take, but it appears certain that the external pressures will be at least as great as the internal ones. However, such changes due to interdependencies among the parts of the system are not to be disdained. After all, as the other components become more powerful, the parser might even become simpler.

IV THE LANGUAGE DEFINITION

Prepared by Jane J. Robinson

Contents:

A. Introduction

In this section we describe the subset of natural English defined for our speech understanding system and the syntax of the protocols that influenced our selection of vocabulary and phrase types. The description includes a discussion of the limitations of the language, the criteria for choosing directions in which to extend it, and the extensions that are currently being developed. A detailed examination of some of the definitions follows, with emphasis on the syntax-oriented factor statements that influence the parser in focusing on one of several competing alternative definitions to apply to an utterance.

Before proceeding, however, a brief review of terminology is in order, with some explanation of how terms are to be interpreted.

It is customary to call the component that defines the subset of English for a speech understanding system a 'grammar'. We feel that 'grammar' is too exclusively associated in most people's minds with syntax or with generative rules where 'levels' of phonology, morphology, syntax, and semantics are more or less rigidly stratified, or with rules that assign degrees of grammaticality to possibly deviant utterances. For reasons that should become clear, we want to avoid these associations. We prefer to call the system component a 'language definition'.

Briefly, our language definition has two major parts:

(1) A collection of word definitions, where 'word' means an unanalyzed, elementary unit.

(2) A collection of composition-rule definitions for combining words and phrases into larger units.

A definition of either kind is 'applied' to some portion of an utterance by the focused parser. The portion may contain gaps and some of its attributes may be unknown, in which case some of it is 'undefined'. The portion receives a score as an 'instance' of the definition. The definition may apply to it quite well, or satisfactorily, or badly, or not at all.

Some definitions apply more often under some circumstances than under others. For instance, definitions for interrogative words and phrases are more likely to apply to utterances in which the speaker is querying a data base than they are to utterances in which he is giving instructions. On the other hand, an utterance is more likely to be declarative than interrogative if the utterance that precedes it was a request for information. Facts like these can be part of the definitions; that is, the definitions can be 'tuned' to a task or discourse situation.

Word definitions and composition-rule definitions are not stratified; they overlap. A given category label may be instantiated both by words and by phrases. The category of whole utterances, U, is a category for words like "Okay" and also for sentences. NP is instantiated by the phrase "the submarine" and also by the word "it".

A definition can reference any source of information--acoustic, phonetic, phonological, prosodic, syntactic, semantic, and pragmatic--to determine the attributes of a proposed instance to which the definition is assumed to apply. These attributes are also referenced in a part of the definition called 'factors', which specifies acceptable, unacceptable, favorable, and unfavorable combinations of attribute values.

When all the attributes are defined and all the factors computed from them are favorable, confidence in the applicability of the definition is high, but this is not to say that less well-defined utterances are ungrammatical or deviant. For example, if an utterance begins with "Does it ..." and ends with a rising pitch, it fits the definition of a yes/no question better than if only one of those attributes is present. On the other hand, the utterance "It has a speed of 20 knots?", uttered with rising pitch and without inverted word order, may be quite clearly understood as asking a question. Language is a redundant system. Utterances may lack some signals without being misunderstood.

B. The Use of Protocol Analysis

1. The Data Management Protocols

In choosing the initial set of words and constructions for our defined language, we were guided by an analysis of the first data management protocols we collected from two Navy officers at the Naval Postgraduate School (Subjects A and C), and to some extent by a pseudo-protocol in which a speaker (Subject B) was directed to imitate them. In no case were speakers asked to limit themselves in their use of English.

In analyzing the results, we were aided by a key-word-in-context concordance program (KWICO) developed at SDC by Georgette Silva. A sample of a merged concordance of the utterances of all three subjects appears in Figure IV-1. The number at the right of each entry is the protocol utterance number. There were approximately 220 utterances in all, using approximately 1800 tokens, representing approximately 170 types of

PAGE 014

```
                         HOW MANY MISSILE LAUNCHERS?                                                       002043
                          NUMBER OF MISSILE LAUNCHERS?                                                     003015
ICH SUBMARINES HAS THE GREATEST NUMBER OF MISSILE LAUNCHERS?                                               003037
S OF SUBMARINE HAS THE GREATEST NUMBER OF MISSILE LAUNCHERS?                                               003038
WHAT SUBMARINE HAS THE GREATEST NUMBER OF MISSILE LAUNCHERS?                                               003040
AT, HOW MANY SUBMAR UNITED STAT F SUBMARINES HAVE LENGTH LESS THAN THIRTY FEET.                            003063
                            WHAT IS THE LENGTH OF A GEORGE WASHINGTON NUCLEAR SUBMARINE?                   001022
                            WHAT IS THE LENGTH OF A LAFAYETTE SUBMARINE?                                   001005
                            WHAT IS THE LENGTH OF AN ETHAN ALLEN SUBMARINE?                                001014
                        WHAT IS THE LENGTH OF THE ETHAN ALLEN?                                             003011
                                AND THE LENGTH OF THE GEORGE WASHINGTON?                                   003012
                             WHAT'S THE LENGTH OF THE GUPPY III AND THE GUPPY II?                          002063
                            WHAT IS THE LENGTH OF THE LAFAYETTE?                                           003004
THE SURFACE DISPLACEMENT, SUBMERGED DISPLACEMENT, LENGTH, NUMBER OF TORPEDO TUBES, NUMBER OF REACTORS,  MBER OF MI SILE 003002
                                         THE LENGTH?                                                       002005
                            WHAT IS THE LENGTH?                                                            002013
                                              LENGTH?                                                      002021
ANY SUBMARINE IN YOUR DATA BASE WHO HAS A BEAM OF LESS THAN OR EQUAL TO 30 FEET.                           001080
ALL SUB ALL UNITED STATES SUBMARINES WITH AH BEAM LESS THAN THIRTY FEET.                                   003062
 MANY SUBMAR UNITED STATES SUBMARINES HAVE LENGTH LESS THAN THIRTY FEET.                                   003063
BSET ON ALL UNITED STATES SUBMARINES AH WITH BEAM LESS THAN THIRTY FEET.                                   003064
                  LIST ALL SUBMARINES WITH BEAM LESS THAN THIRTY FEET.                                     003065
                               I'M SORRY, LESS THAN 30 FEET AND HAS A.                                     001081
E SUBMARINES IN YOUR DATA BASE WHO HAVE A BEAM OF LESS THAN 30 FEET AND HAVE A SUBMERGED SPEED OF GREATER THAN 2  KNOTS, 001083
L THE SUBMARINES IN YOUR DATA BASE WITH A BEAM OF LESS THAN 30 FEET, WHAT IS THE SPEED OF THE FASTEST?     001067
 ALL THE SUBMARINES IN YOUR DATA BASE WITH A BEAM LESS THAN 30 FEET, WHAT IS THE FASTEST SUBMERGED SPEED?  001087
    WHAT CLASSES OF THE US ATTACK BOATS HAVE BEAMS LESS THAN 30 FEET?                                      002051
                                            LET ME RESTATE THE QUESTION.                                   001082
                              REPEAT THE LIST AGAIN.                                                       003067
                                         LIST ALL CLASSES OF OF NUCLEAR BALLISTIC MISSILE UNITED STATES SUBMARI 003001
                                         LIST ALL SUB ALL UNITED STATES SUBMARINES WITH AH BEAM LESS THAN THIRT 003062
                                         LIST ALL SUBMARINES WITH BEAM LESS THAN THIRTY FEET.              003065
                             GIVE ME THAT LIST OF SUBMARINES AGAIN.                                        003044
                              REPEAT THAT LIST.                                                            003066
                                  AND HOW MANY AH DIESEL PATROL SUBMARINES DO THEY HAVE?                   003032
                                  AND HOW MANY AH DIESEL SUBMARINES AH DO THE BRITISH HAVE AH DAH AH.      003033
                                      HOW MANY AH NUCLEAR ATTACK SUBMARINES DOES THE UNITED STATES HAVE?   003022
                                      HOW MANY ATTACK NUCLEAR SUBMARINES DOES TH  UNITED STATES HAVE?      001039
                                      HOW MANY CRUISE MISSILE NUCLEAR SUBMARINE    ES RUSSIA HAVE?         001040
                                  HOW HOW MANY DIESEL GUIDED MISSILE SUBMARINES AND HOW MANY DIESEL TRAINING SUB 003025
                                      HOW MANY DIESEL GUIDED MISSILE SUBMARINES DOES THE UNITED STATES HAVE? 003028
                                      HOW MANY DIESEL GUIDED MISSILE SUBMARINES DO THE SOVIETS HAVE?       003031
                                      HOW MANY DIESEL PATROL SUBMARINES DO THE BRITISH HAVE?               003034
                                      HOW MANY DIESEL PATROL SUBMARINES DO THE RUSSIANS HAVE?              003035
                                      HOW MANY DIESEL SUBMARINES DOES THE UNITED STATES HAVE?              003030
                                      HOW MANY DIESEL TRAINING SUBMARINES DOES THE UNITED STATES HAVE?     001043
HOW MANY DIESEL GUIDED MISSILE SUBMARINES AND HOW MANY DIESEL TRAINING SUBMARINES DOES THE UNITED STATES HAVE? 003025
                                   AH HOW MANY DIESEL TRAINING SUBMARINES DOES THE UNITED STATES HAVE?     003029
                                      HOW MANY ETHAN ALLEN CLASS NUCLEAR SUBMARINES DOES THE UNITED STATES NAVY 001018
                                  AND HOW MANY ETHAN ALLENS ARE THERE?                                     003016
                                      HOW MANY FLEET NUCLEAR SUBMARINES DOES BRITAIN HAVE?                 001041
                                      HOW MANY GEORGE WASHINGTON CLASS NUCLEAR SUBMARINES DOES THE UNITED STATES 001026
                                  AND HOW MANY GEORGE WASHINGTONS?                                         003017
                                      HOW MANY GUIDED MISSILE DIESEL SUBMARINES DOES THE UNITED STATES HAVE? 001042
                                      HOW MANY GUIDED MISSILE DIESEL SUBMARINES DOES RUSSIA HAVE?          001044
```

FIGURE IV-1 SAMPLE CONCORDANCE PAGE

stems and affixes. Most of the words that appeared more than once are included in our word definitions, although we are not equally satisfied with all of them.

From the concordance we can see quickly which words occurred frequently and can determine how (in what contexts) a given word was used. The concordance shows that "list" occurs six times--three times as a verb and three times as a noun--and that "many" is always preceded by "how". The concordance also helps us spot those phrases that are marked by the presence of tokens from a small set of word types, that is, by what are often called 'function words'. For example, there were 11 tokens of "that" of which three functioned as relative pronouns. All three occurrences of "who" were relatives, and "which" occurred twice as a relative out of a total of ten occurrences. Altogether there were 24 occurrences of these words, eight of them as relatives. This accounted for all the relative clauses in the protocols; there were none with suppressed relatives. The utterance numbers in the concordance allow comparisons of the frequencies with which the three speakers used various words and phrases. We learned, for instance, that all eight relative clauses came from a single speaker, Subject A.

As expected, given the nature of the problem they were assigned, most utterances from the two Navy subjects were WH interrogatives, with two types predominating: "What is the X of the Y?" (60 utterances out of 147), and "How many Xs does the Y have?" (25 occurrences). Yes/no interrogatives and imperatives were rare, and there was only one declarative. In our current language definitions, we have defined different likelihoods for the different sentence types on the basis of these observed differences in frequency. This is a way of 'tuning' the language definition to a type of discourse. It can be tuned for a different type quite easily, but we have not yet evaluated the effects of tuning.

Subjects A and C differed strikingly with respect to their use of ellipsis. A, the one who used relative clauses, made no use of ellipsis. Thirty-one of C's 61 utterances were elliptical nominal expressions. Also, while A used superlatives 12 times and comparatives five, C used no superlatives and only one comparative. If speakers querying the same data base for the same purpose turn out to diverge widely and consistently with respect to the use of some syntactic categories, we will define user-dependent attributes for those categories and will provide easy mechanisms for switching from one user profile to another.

2. Comparison with the Computer Consultant Protocols

Since we want to define a language that is adequate for more than one task domain, we are analyzing the similarities and differences (in addition to domain-dictated differences in vocabulary) between the data management protocols and some of the computer consultant dialogs that are being collected in conjunction with the Computer Based Consultant Project at SRI

(Nilsson et al., 1974, 1975; Hart, 1975). These dialogs concern the maintenance of electromechanical equipment in a workstation environment with the system acting as a consultant. Concordances have been made for them as well, using an SRI program.

One interesting difference that emerged from the comparison was in the use of negatives. In the computer consultant task, the user and the consultant (system) have to share information and arrive at a common picture of the world from time to time. The consultant has to ask whether some state-of-affairs holds; the user has to ask what state-of-affairs should be sought. These needs give rise to yes/no questions and--since the answers are sometimes "no"--to negatives. ("Yesses" outnumbered "noes", but many "yesses" were assents to directions given by the consultant rather than answers to questions; e.g., "Fasten the pump belt." "Yes. What tool should I use?") Also, the workstation task entailed trouble-shooting, and trouble-shooting is a situation in which negative information is useful. "There doesn't appear to be any pressure." "There is no on-off switch; if anything goes wrong just ..." These dialogs contained over 80 negative tokens of various types (about 1%), the reduced "-n't" form being the most common.

Somewhat unexpectedly, there were no negatives in the data management protocols, possibly because the problem assigned to the speakers was specified completely at the outset. We will shortly begin collecting new data management protocols in which the subjects, while still querying a static data base, will be presented with a scenario that provides additional information at selected intervals. It will be interesting to see if the incidence of yes/no questions and negatives increases under these circumstances.

3. Comparisons with General English

Some of the similarities between the two sets of protocols are predictable from the characteristics of General English. Inspection of the concordances showed that while the frequencies for "submarine" and "bolt" were domain dependent, frequencies for the preposition "of" were much alike. Word-frequency tables for English consistently rank "of" among the top three or four words, and in most texts, "of" prepositional phrases are among the favored modifiers of head nouns. For instance, they are preferred above preposed genitives for denoting inalienable possession, except when the possessor is denoted by a pronoun; that is "the speed of the Lafayette" and "the base of the pump" are more likely than "the Lafayette's speed" or "the pump's base", but "its speed" and "its base" are more likely than "the speed of it" or "the base of it".

Prepositional-phrase modifiers of nouns were far more common than relative clauses in both sets of protocols. This seems to be a characteristic of General English also, especially of spoken English, although statistics for a large amount of spoken English are not available for checking this impression.

(It is obviously easier to count tokens of a word type, since all have the same form; however, tokens of phrase types are difficult to spot unless all of them are marked by tokens from a small number of types. Since relative clauses may be unmarked, as in "the tool you need", it takes careful and time-consuming scrutiny of text to catch them.) The scarcity of relative clauses in the submarine protocols has already been noted. Of 95 "thats" in the workstation dialogs, only 11 introduced relative clauses; only six of the 23 "whiches" did so; and the lone "who" was an interrogative. No attempt was made to establish how many relative clauses not introduced by a relative word occurred in the workstation dialogs, but scanning indicated they were not numerous.

4. Limits and Extensions

We included composition-rule definitions for "of" phrases very early in our definition language, both because they are so ubiquitous and because it seemed doubtful that speakers could observe an instruction to avoid using "of" while speaking naturally in going about their tasks. While it might be possible for them to avoid using relative clauses, we are adding them in the interests of generality. Some new phrase compositions will have to be added, but some relative clause types have compositions that have already been defined for the category S. In these cases, extension entails adding new attributes and factors.

At present about 60 phrase types and 30 categories have been defined for testing with acoustic input. These include elliptical utterances, ten major varieties of sentences, all major spoken integer expressions, "be" and "do" auxiliaries, 15 major noun-phrase types, transitive, intransitive and passive verb phrases, prepositional phrases, genitives, negatives, and limited comparisons. Clearly, the defined language is a very small subset of English. Its syntax is possibly adequate for the first data management protocols; probably not for the new ones being gathered; certainly not for the computer consultant dialogs. Some of the definitions are not as complete as others. Some could make good use of prosodic constraints, but we do not yet know how to supply them, although the structures for handling them are provided. Of the two kinds of limitations that we have, limitations of coverage and limitations of depth, we feel it is better to develop the definitions we have before we attempt broad coverage.

Nevertheless, some additional coverage is desirable now. In choosing among possibilities, we are guided by several criteria. We aim to develop definitions of words and phrases that are typical of discourse required for the tasks, that are also representative of significant syntactic/semantic problems, and that are sufficiently tractable to be put into the system and tested within a reasonable time.

An immediate extension is to constructions in which a noun is modified by a preceding noun-stem. People are prone to call these noun-stems 'adjectives', but the difference between the two is clear when one compares "dirt floor" with "dirty floor" or "Lafayette class submarine" with "large yellow submarine". That they are stems rather than nouns is shown by the absence of the requirement for number agreement in constructions like "a 30-foot beam" and "a three submarine task force". ("Parts list" and "operations research" are among the few exceptions to the generalization that plural noun forms do not premodify nouns. Exceptions are best treated in the lexicon.)

Figures on the relative frequency of adjective and noun-stem modifiers are hard to obtain. However, a scanning of the protocols and of conversations and texts picked at random gives the impression that noun-stem modification is far more frequent than modification by attributive adjective. (A notice on the nearest bulletin board concerns a "day hike" starting at "park headquarters", and there is not a single adjective in the paragraph.) Often the noun-stems are so closely linked to a following noun that it seems best to make the two into a single lexical entry. We have followed this policy with respect to compounds like "setscrew" and "torpedo tube". For expressions like "Lafayette class" such a policy would be unwise, since every name in the data base can enter into such a construction. Not only that, the process is recursive, as shown by "Lafayette class submarine".

In our definitions, a noun stem, N, does not have a number attribute. It is neither singular nor plural. NOUNs do have the attribute. Singular NOUNs are not distinguishable phonemically from Ns, but a NOUN generally has a different prosody from an N. We have noticed in our protocols, for instance, that "United States" is generally shorter and the last syllable more reduced when it occurs as an N, as in "United States Navy" than when it is a NOUN, as in "the navy of the United States". This is no doubt a secondary effect; as a modifier (N) it cannot be at the end of a phrase. Ns and NOUNs will have distinctive prosodic attributes in our definitions.

Among constructions that pose severe problems are those containing coordinate conjunctions. We are including them because they are needed for natural discourse, and also because they are representative of significant problems concerning scope and parallelism. We are not attempting to treat them with full generality, but are selecting from among the different kinds of coordinate constructions some that are both representative and that we can handle relatively easily. These, specifically, are coordinate noun-phrases and nominals, e.g., "nuts and bolts". We think it should not be unreasonably hard for speakers to avoid some of the frequent uses of "and", in particular those uses in which it is a 'sequencing word', replaceable by "then", e.g., "Loosen the motor bolts and slide the motor ..." In the same limited fashion, we are extending our defined language to include more comparative constructions, quantifiers, additional kinds of number expressions, and superlatives.

The computer consultant dialogs made much use of the indexical "I/you" expressions. These were accompanied by frequent use of modals, verbs of knowing and sensing, imperatives, and embedded complements, as in "I would like to know if ..." and deontic anxieties like "How tight should I make it?" and "Be sure to ..." Many of these complex constructions are circumlocutions that can be omitted without loss of pragmatic meaning. "I would like to know" is not necessary for asking a question. Other constructions are central to the task. "What are you doing now?", "Did you ...?", and "What do I do next?" are natural to the changing world of the workstation environment. We are now extending the definitions to include tense, progressive aspect, and expressions with sequencing words like "first", "then", and "next". A few modals--"can", "should", and the ubiquitous "have to" ("hafta")--are planned for later. These are to cover expressions like "How can I ...?", "Where should it go?", and "How tight does it hafta be?". Questions like these and the sequencing expressions referred to above imply a model of the task that motivates the discourse of which they are a part. It is not possible to "understand" the question "What do/should I do next?" without it. A task model is being developed by the Computer-Based Consultant Project at SRI. Definitions for our subset of English will reference it.

C. Performance Syntax

1. Orientation

This section presents, informally, some of the syntax-oriented parts of the word and composition-rule definitions.[1] To provide an overview, Figure IV-2 lists the word categories and Figure IV-3 the phrase compositions. Examples accompany the entries in each list. More complete versions of the definitions appear in Appendix A. The examples that accompany the individual definitions given there show instances to which the definition applies normally or especially well or poorly or badly or not at all. Here we examine and explain some of the statements in the definitions, especially the factors.

How the factor statements are evaluated and used is explained in Section III, The Parsing System. Briefly, factors reference the attributes exhibited by some instance--or some purported or predicted instance--to which the rule definition in question might apply, in order to judge the degree of applicability of the rule to the instance. If one factor lowers the score for applicability, others may raise it. For example, "which is what" is judged less likely than "which is that" by one factor that lowers the score for applying the composition rule S = NP AUXB NP to it. However, if "which is what" is actually uttered, there may be acoustic, prosodic, semantic, and discourse

------

[1] A discussion of the semantics-oriented parts of the word and composition-rule definitions is provided in Section V, Semantics.

factors that enhance its overall score. The overall score is used by the parser to choose among competing parsings. A low score may eliminate a parsing path; a high one may raise the priority of a parsing path.

Figure IV-2 Word Categories in the Language Definition

CATEGORY	EXAMPLES
N(noun stem)	Lafayette, Ethan.Allen, speed, submerged, speed, draft, knot, foot
NOUN	feet
NP	I, you, it, we, us, they, them, who, whom
DET	that, those, this, these, which, what
ART	a, the
DIGIT	one, two, twen, three, thir, four, five, fif, ... nine
TEEN	ten, eleven, twelve
BIGNUM	hundred, thousand, million, billion
BE	am, is, are
DO	do, does, dont
V(verb stem)	list, own
VERB	has, have
PREP	of, by, with
QUANT	all, some, any
MP	few, little, many, much
NUMBERP	how,many
THANR	more, less
U	okay

Figure IV-3 Composition Rules in the Language Definition

NAME	COMPOSITION	EXAMPLES*
U1	U=S	what is the surface displacement of the Lafayette
U2	U=NP	the Ethan Allen
U3	U=NOM	submerged displacement
S1	S=NP VP	we/have it
S2	S=NP AUXD VP	we/don't/have it
S3	S=NP:NP1 AUXB NP:NP2	what/is/it
S4	S=VP	list it
S5	S=AUXD VP	don't/list it
S6	S=NP:NP1 AUXD NP:NP2 VP	what/do/we/have
S7	S=AUXD NP VP	do/we/have them
S8	S=AUXB NP:NP1 NP:NP2	is/it/a submarine
S9	S=NP AUXB VP	it/is/owned by the Russians
S10	S=AUXB NP VP	is/it/listed
NP1	NP=NOM	fuel, submarines
NP2	NP=NUMBER	how much, twenty, more than four
NP3	NP=NUMBERP"OF NP	more than four/of/them
NP4	NP=NUMBERP NOM	twenty/knots, more than two/subs, how many/ships
NP5	NP=DET	which, those
NP6	NP=DET "OF NP	which/of/them
NP7	NP=DET NOM	those/subs
NP8	NP=DET NUMBER "OF NP	which/two/of/them
NP9	NP=DET NUMBER NOM	those/two/ships
NP10	NP=DET NUMBER	this/one
NP11	NP=ART NOM	a/ship, the/fuel
NP12	NP=ART NUMBER "OF NP	a/hundred/of/them
NP13	NP=ART NUMBER NOM	a/hundred/ships
NP14	NP=ART NUMBER	a/thousand
NOM1	NOM=NOMHEAD	submerged,speed
NH1	NOMHEAD=NOMHEAD PREPP	submerged speed/of the Lafayette
NH2	NOMHEAD=NOUN	feet, Lafayettes, speed
N1	NOUN=N	foot, Lafayette
N2	NOUN=N -PL	Lafayette/-s, submarine/-s
VP1	VP=VERB	list
VP2	VP=VP NP	list/them
VP3	VP=VP PREPP	owned/by the Russians
V1	VERB=V	list
V2	VERB=V -SG	list/ -s
V3	VERB=V -PPL	list/ -ed

*Slashes separate the constituents identified in the composition rules.

Figure IV-3 Composition Rules in the Language Definition (concluded)

NAME	COMPOSITION	EXAMPLES*
D1	AUXD=DO	do, does, don't
D2	AUXD=DO NEG	do/not
D3	AUXD=DO -NT	does/ -n't
B1	AUXB=BE	is, are, am
B2	AUXB=BE NOT	is/not, are/not, am/not
B3	AUXB=BE -NT	is/ -n't, are/ -n't
PREPP1	PREPP=PREP NP	of/the Lafayette
DET1	DET=NP -GEN	the Lafayette/ -'s
NUMP1	NUMBERP=HOW MP	how/much
NUMP2	NUMBERP=MP	many, much
NUMP3	NUMBERP=THANR "THAN NUMBER	more/than/four
NUMP4	NUMBERP=NUMBER	four
NUM1	NUMBER=SMALLNUM	one, fifty one, ten
NUM2	NUMBER=BIGADD	four hundred and fifty one
NUM3	NUMBER=BIGMULT	fifty one thousand
NUM4	BIGMULT=SMALLNUM BIGCAT	fifty one/thousand
NUM5	BIGMULT=BIGADD BIGCAT	four hundred and fifty one/ thousand
NUM6	BIGMULT=BIGCAT	hundred, thousand
NUM7	SMALLNUM=DIGIT	one, five, fif, nine
NUM8	SMALLNUM=TEEN	fifteen, nineteen
NUM9	TEEN=DIGIT -TEEN	fif/-teen, nine/-teen
NUM10	SMALLNUM=DIGTY	fifty, ninety
NUM11	DIGTY=DIGIT -TY	fif/-ty, nine/-ty
NUM12	SMALLNUM=DIGTY DIGIT	fifty/two
NUM13	BIGADD=BIGMULT SMALLNUM	four hundred/fifty two
NUM14	BIGADD=BIGMULT "AND SMALLNUM	four hundred/and/fifty two
NUM15	BIGADD=BIGMULT BIGADD	fifty thousand/four hundred and fifty two
NUM16	BIGMULT=BIGMULT BIGCAT	four hundred/thousand

*Slashes separate the constituents identified in the composition rules.

Our mnemonic terms for factor scores are VERYGOOD, GOOD, OK, POOR, BAD, and OUT. These are not meant to be judgments of grammaticality or acceptability, but rather to be expressions of an estimate of likelihood. They are necessarily vague, because we are dealing with gradual phenomena, probabilistic tendencies, and vacillating intuitions. (The last source of vagueness should decrease as more protocols are studied.) They mean something like "quite likely", "frequent", "ordinary", "odd but possible", "unlikely--listen again", and "so special that we do not expect it in our task domain and do not define it". This is not to claim that a phrase or composition is excellent or wrong or absolutely impossible. Some compositions, like "foot" with "-s" as a plural noun, are indeed wrong in English and OUT for our subset of English. On the other hand, "fuel" does combine with "-s" to form a plural noun in English, with the specialized meaning "kinds of fuel". For the time being, "fuels", like "foots" is judged to be OUT for our language. This judgment may be altered. If we find that our language users refer to kinds of fuel as "fuels" ordinarily or often, then the judgment OUT will be changed to OK or even GOOD.

2. Word Categories and Attributes

A high-frequency syntactic pattern in our submarine protocols is one in which a WH noun phrase is followed by a form of "be", followed by another noun phrase. An instance is:

What is the surface displacement of the Lafayette

Variants of an appropriate answer include:

Seven thousand tons

The surface displacement of the Lafayette is seven thousand tons

Lafayettes have a surface displacement of seven thousand tons

The question and its answers contain tokens of three words: "Lafayette", "surface displacement", and "ton". Although all three belong to the same category N, or noun stem, they have quite different semantic attributes. "Lafayette" denotes a class or a member of a class of concrete, countable objects. "Surface displacement" is an abstract relational noun stem, denoting a relationship between a concrete object and the amount or weight of water it displaces when it is on the surface. The noun stem "water" is not in the question or in its answers, but "ton" denotes a unit suitable for measuring materials like water and fuel that are not discrete objects, and therefore not countable.

The underlying semantic differences in the three kinds of noun stems affect their combinability not only with each other and with verbs and adjectives but also with plural suffixes, words of small closed classes such as the determiners, "this", "these", "that", "which", and the articles, "a" and "the", as well as with number expressions like "seven thousand". It would seem odd for someone to say:

That surface displacement of the Lafayette

The surface displacement of these seven thousand tons

Which seven thousand tons of those surface displacements

Similarly, "two surface displacements" is at least as peculiar as "two fuels", although for a different reason. Moreover, while

The submarine is Lafayettes

is ungrammatical,

The surface displacement is seven thousand tons

seems fairly ordinary, although it is built on the same syntactic pattern: singular noun phrase / "is" / plural noun phrase.

The semantic distinctions arising from the different denotations of the noun stems show up at the surfaces of utterances, where we may consider them to be syntactic constraints. Syntactic constraints are coarser than semantic ones, making them easier to process at a superficial, less costly level. Therefore, we have specified them in the attributes and factor statements of the definitions even where they repeat material appearing in the semantic component. This makes them available to the parser for judging among alternatives before having to obtain full semantic and acoustic information. For instance, if someone says "those fuel supplies", we do not want the parser to explore in depth the application of rules that build a plural noun-phrase from "those fuel s..." without considering an alternative definition in which "fuel" is a modifier of a countable nominal beginning with "s". To this end, we include a factor statement that checks the countableness of "fuel" by referencing its count/mass/unit (CMU) attribute.

Figure IV-4 shows some of the 'syntactic' (syntax-oriented) attributes defined for noun stems, approximately as they appear in the lexicon. Other stems resembling "Lafayette" are those that denote other submarine classes and the stem "submarine" itself, as well as "missile launcher", "torpedo tube", etc. Those resembling "surface displacement" include "speed", "beam", "draft" and "length" and, of course, "submerged displacement". Those resembling "ton" and "foot" include "knot".

Figure IV-4 Syntactic Attributes for Noun Stems

```
WORDS.DEF     N

    FUEL
        CMU = (MASS);

    FOOT
        CMU = (UNIT),
        PLSUFF = NO;

    LAFAYETTE;

    SURFACE.DISPLACEMENT
        RELN = T;

    TON
        CMU = (UNIT);
```

One way in which Ns may differ from one another is in the set of values they may have for the CMU (Count-Mass- Unit) attribute. The first entry in Figure IV-4 defines the value of the attribute for "fuel" as (MASS). The values for "foot" and "ton" are both (UNIT). "Lafayette" and "surface displacement" are not marked for the attribute; however, a redundancy function assigns the value CMU = (COUNT) to all members of the category N if the attribute is not otherwise specified in their individual entries. General facts of this kind are more efficiently stated as category attributes--or factors, as the case may be--when large numbers of entries are affected. The attribute RELN = T, which marks "surface displacement" as a relational noun-stem, differentiates it from the other entries, while "foot" is distinguished by the attribute PLSUFF = NO, which marks it as not taking the regular plural ending. Only this last category is purely syntactic. The RELN attribute is basically semantic. (Its interpretation is discussed in Section V, Semantics.) Syntactically, relational noun stems do not combine readily with plural suffixes and number expressions. When they do combine, the semantic structures are specialized. An example is "three speeds", meaning three rates of speed. "Three fuels", meaning three kinds of fuel, is analogous. To some degree, relational noun stems may share with mass noun stems the property of nondiscreteness. However, "a speed of twenty knots" is ordinary, while "a fuel of two tons" is ill-formed.

The syntactic properties of nominal expressions headed by relational noun stems are highly dependent on which aspect of the relation is referred to in an accompanying prepositional phrase. As shown in more detail later, the same composition rule applied to different instances such as "draft of the Guppy II" and "draft of five feet" defines different attributes for them. The difference in attributes marks the syntactic fact that the two instances do not fit with equal ease in all syntactic environments; for example, consider the two environments:

it is the ....

and

it has a ....

The two environments are quite different syntactically but not very different phonetically. If the only firm acoustic anchor points given to the parser are the stressed nominals "draft" and a following "Guppy II", it should be influenced to choose the first environment as the better alternative. (Compare "it is the draft of the Guppy II" with *"it has a draft of the Guppy II".) If the anchor points are "draft" and "five feet", the second is better. (Compare "it has a draft of five feet" with ?"it is the draft of five feet".)

Noun stems with the CMU value UNIT also exhibit special syntactic behavior. They combine easily with plural suffixes and number expressions (e.g., "two knots", "five feet"), but not so well with definite determiners ("those two knots"). Also noun phrases in which they are heads do not combine with genitive suffixes. Thus, while "the Ethan Allen's speed" is ordinary, "the twenty knots' speed" is ill-formed.

In the next section, the effects of these initial attributes of noun stems, and of some other forms as well, are traced through successive composition rules.

3. Composition Rules, Attributes, and Factors

The attributes of the Ns that affect their ability to combine with the plural suffix "-s" are referenced in the two composition rules, N1 and N2, defining the category NOUN. These appear in Figure IV-5.

The attribute statements propagate or 'bubble up' the attributes of the stem constituent, adding a number attribute (NBR) that is singular (SG) for N1 and plural (PL) for N2. The first factor statement of N1 references the CMU attribute assigned to the noun by the attribute statement and states that if the value is MASS, then the score is enhanced. It is GOOD. This judgment incorporates our knowledge that the other rule, N2, in which N is a constituent, cannot apply to mass noun-stems. Therefore, if the token to which the composition rule is applying is a mass noun-stem, this is the right composition rule to apply. When new composition rules are defined in which Ns premodify a nominal, the possibilities and expectations may be altered, but we will still want to enhance the score of N1 for mass Ns.

Figure IV-5 Portions of Composition-Rule Definitions for NOUNs

```
RULE.DEF N1      NOUN = N;
    ATTRIBUTES
        CMU,RELN,PLSUFF FROM N,
        NBR = "(SG);
    FACTORS
        CMU = IF CMU EQ "(MASS) THEN GOOD ELSE OK,
        RELN = IF RELN EQ "T THEN GOOD ELSE OK,
        IF PLSUFF = NO THEN GOOD ELSE OK;
    EXAMPLES
        FUEL (GOOD)
        SURFACE DISPLACEMENT (GOOD)
        FOOT (GOOD)
        TORPEDO TUBE (OK);

RULE.DEF N2      NOUN = N -PL;
    ATTRIBUTES
        CMU,RELN,PLSUFF FROM N,
        NBR = "(PL);
    FACTORS
        PLSUFF = IF PLSUFF EQ "NO THEN OUT ELSE OK,
        CMU = IF CMU EQ "(MASS) THEN OUT ELSE OK,
        UNIT = IF "UNIT IN CMU THEN GOOD ELSE OK,
        RELN = IF RELN EQ "T THEN POOR
                 ELSE OK;
    EXAMPLES
        FOOT -S (OUT)
        FUEL -S (OUT)
        SURFACE DISPLACEMENTS (POOR)
        TONS (GOOD)
        SUBMARINES (OK);
```

Similar reasoning motivates the RELN factor statement. However, the relationship between the factor statements of the two composition rules is different for this attribute. The relational noun attribute enhances the application of N1 but does not block the application of N2, whereas the mass attribute enhances the application of N1 and blocks the application of N2. (These judgments are admittedly subtle and tuning them to each other is a nontrivial task.)

The plural suffix factor statements (PLSUFF), like the CMU statements, enhance the score for applying N1 to stems that do not take a plural suffix and constrain N2 not to apply to them. The UNIT factor of N2 enhances the score when the composition rule applies to an N with the value UNIT in its CMU attribute. This judgment is based on the fact that, in our current task domain, all the measured properties have measurements exceeding one unit and on the reasonable expectation that less than two units is a special case.

The attributes of the Ns continue to be propagated through successive composition rules so that noun phrases acquire the attributes, with exceptions and some additions, of the Ns that are their heads.

One of the added attributes of noun phrases is FOCUS. A noun phrase is definite (DEF) if its first immediate constituent is the definite article "the" or one of the determiners: "this", "that", "these", "those", "what", "which". It is indefinite (INDEF) if the article is "a" or if there is no article or determiner constituent. Noun phrases that are also proper names (e.g., "Russia") are exceptions. They are marked DEF in the lexicon.

Combining definite focus with a nominal headed by a word denoting a unit is unusual. Compared with "which torpedo tube", "which seven thousand tons" seems odd. So does "a draft of the five feet" compared with "a submarine of the U.S. fleet" and "those twenty knots" compared with "those missile launchers". Indefinite focus is more common for units: "a ton", "a draft of five feet", "twenty knots". It does not imply a uniquely determinable object or set of objects, pointed to in the discourse. Units are seldom talked about per se, although it is possible to do so in definitional statements like "the pound is a unit of weight". (There are also expressions like "those extra five pounds", referring to the weight in terms of the units, with a definite determiner.)

Figure IV-6 shows how this tendency is handled in three of the noun phrase composition rules. NP4 defines indefinite noun phrases whose first constituent is a number expression followed by a nominal. NP7 defines definite noun phrases. NP11 defines phrases whose focus may be either, depending on the article found or predicted for the instance.

In each definition, a factor called UNIT references the CMU attribute to judge the applicability of the composition rule to the instance. The CMU attribute is assigned by intersecting the attribute values of the constituents to resolve possible ambiguities. Number expressions have the set of values (COUNT UNIT), except for those containing "much", where the value is (MASS), and those containing "more", where the value is (COUNT MASS UNIT).

Figure IV-6 Portions of Composition-Rule Definitions for Definite and Indefinite Noun Phrases

```
RULE.DEF NP4     NP = NUMBERP NOM;
    ATTRIBUTES
        FOCUS = "INDEF,
        MOOD,NUM FROM NUMBERP,
        NBR = GINTERSECT(NBR(NUMBERP),NBR(NOM)),
        RELN FROM NOM,
        CMU = GINTERSECT(CMU(NUMBERP),CMU(NOM));
    FACTORS
        CMU = SELECTQ CMU WHEN NIL THEN OUT,
        HUN = IF FSTWD(NUMBERP) IN "(HUNDRED THOUSAND MILLION)
            THEN OUT,
        NBR = SELECTQ NBR WHEN NIL THEN OUT,
        UNIT = IF "UNIT IN CMU THEN VERYGOOD ELSE OK,
        RELN = IF RELN EQ T THEN OUT ELSE OK,
        SUBCAT = SELECTQ SUBCAT(NOM) WHEN PROPN THEN OUT;
    EXAMPLES
        FIVE FUELS (OUT)
        HOW MUCH SUBMARINE (OUT)
        ONE SUBMARINES (OUT)
        HOW MANY FUEL (OUT)
        FIVE FEET (VERYGOOD)
        FIVE SUBMERGED SPEEDS OF THREE KNOTS (OUT)
        FIVE SUBMERGED SPEEDS OF THE SUBS (OUT)
        FIVE SUBMARINES (OK);

RULE.DEF NP7     NP = DET NOM;
    ATTRIBUTES
        FOCUS = "DEF,
        CMU = GINTERSECT(CMU(DET),CMU(NOM)),
        NBR = GINTERSECT(NBR(DET),NBR(NOM)),
        RELN FROM NOM,
        MOOD FROM DET;
    FACTORS
        CMUCHK = SELECTQ CMU WHEN NIL THEN OUT ELSE OK,
        UNIT = IF "UNIT IN CMU THEN POOR ELSE OK,
        NBRCHK = SELECTQ NBR WHEN NIL THEN OUT;
    EXAMPLES
        THOSE SUBMARINE (OUT)
        THAT SUBMARINE (OK)
        THOSE FUELS (OUT)
        THAT FUEL (OK)
```

Figure IV-6 Portions of Composition-Rule Definitions for Definite and Indefinite Noun Phrases (concluded)

```
          WHICH TONS (POOR)
          THAT DRAFT OF FIVE FEET (POOR)
          WHAT FUEL (OK)
          WHICH SUBMARINE (OK)
          THAT SPEED (OK)
          THAT SURFACE DISPLACEMENT (OK);

  RULE.DEF NP11     NP = ART NOM;
      ATTRIBUTES
          RELN FROM NOM,
          CMU = GINTERSECT(CMU(ART),CMU(NOM)),
          NBR = GINTERSECT(NBR(ART),NBR(NOM)),
          MOOD = "DEC,
          FOCUS FROM ART;
      FACTORS
          CMU = SELECTQ CMU WHEN NIL THEN OUT,
          NBR = SELECTQ NBR WHEN NIL THEN OUT,
          UNIT = IF "UNIT IN CMU THEN IF FOCUS EQ "DEF
              THEN POOR ELSE GOOD,
          RELN = IF RELN EQ T AND IF FOCUS EQ "INDEF AND
              IF CMU EQ "(COUNT) THEN OUT ELSE OK,
          PROPNCHK = IF SUBCAT(NOM) EQ "PROPN THEN
              [X=FSTWD(ART), IF X EQ "THE THEN GOOD
              ELSE IF X EQ "UNDEFINED THEN OK ELSE OUT]
              ELSE OK;
      EXAMPLES
          A SUBMARINES (OUT)
          THE TON (POOR)
          THE DRAFT OF FIVE FEET (POOR)
          A FUEL (OUT)
          THE SUBMARINE (OK)
          A TON (GOOD)
          A SUBMARINE (OK)
          A DRAFT OF FIVE FEET (GOOD)
          THE SUBMERGED SPEED (OK)
          A DRAFT OF THE LAFAYETTE (OUT);
```

The intersection of the values for "five" and "submarines" is (COUNT); for "five" and "feet", it is (COUNT UNIT); for "much" and "fuel", it is (MASS); and for "much" and "submarine", the intersection is NIL.

If the intersection is NIL, the CMU factor in each composition rule scores the application as OUT. If the UNIT value appears in the CMU attribute after application, then the UNIT factor for NP4 scores the application as VERYGOOD. There are two reasons for this judgment. One is that number expressions are typically found with unit expressions to form measure expressions, and NP4 has a number expression constituent. The other reason is that NP4 defines phrases with indefinite focus, and units are more likely to occur with indefinite than with definite focus, as the preceding examples have indicated.

Since the focus for phrases defined by NP7 is always definite, the UNIT factor decreases the score for applying it when the UNIT value appears in the CMU attribute. For NP11, the UNIT factor judges the application to be GOOD if the article is "a" and UNIT appears in the CMU values, but POOR if the article is "the".

Although NP4 applies especially well to instances in which units are present, it does not apply at all if the head of the nominal constituent is a relational noun stem like "surface displacement" or "speed". In discourse about washing machines and bicycles, "three speeds" might occur in an ordinary way, but for our current discourse, we do not anticipate such a combination.

Certainly, we do not expect "three submerged speeds". Therefore, if the nominal is relational (RELN = T), a factor RELN prevents NP4 from applying.

On occasion, a constraint like the one above may bypass the need for detailed discourse analysis or acoustic mapping to eliminate a wrong parsing path. To see how this effect is achieved, consider the following example. Assume that the acoustic mapper has made some tentative identifications and offered both "submarine" and "submerged speed" as acoustically plausible alternatives for filling the gap in the partially analyzed phrase "three ----- -s of the U.S. Navy". This is not improbable since "submarines" and "submerged speeds" resemble each other in many ways. They both start with "s"; their first syllables have central vowels; their last syllables have high front vowels, and so forth. If NP4 is to be applied, however, the relational noun factor will resolve the doubt in favor of "submarine", and there will be no need to test in depth how well "submerged speed" maps onto the acoustic data or how well it fits the semantic and discourse constraints.

In a somewhat different way, the UNIT factor of NP11 guides the choice between "a" and "the", where acoustic evidence for a choice is typically lacking. In its semantics, "a" resembles the number "one" in its ability to combine with numbers and units; e.g., "one ton", "a ton", "one hundred", "a hundred". If the instance of the nominal constituent is "ton", "foot", "knot", or some other singular expression with the value UNIT for its CMU attribute, the "a" is judged to be more likely than "the" as the form of the article.

On the other hand, if the nominal has "fuel" or "submarines" as its head, the article cannot be "a". The CMU attribute for "a" is (COUNT UNIT), which does not intersect with the value (MASS) of the CMU attribute for "fuel"; the NBR attribute is (SG), which does not intersect with the value (PL) for "submarines". The factors referencing these attributes rule out application when the intersection is NIL. These are typical syntactic agreement tests, used in all three definitions in Figure IV-6 as well as in many other composition-rule definitions.

Attributes derived from instances of noun phrases are also propagated to the prepositional phrases in which they are objects. In effect, a prepositional phrase is a noun phrase with a preclitic that marks its case relationship to another noun phase or to a verb phrase. The term 'case' does not refer to the grammatical case-endings or inflections of pronouns, but refers to semantic relations in the sense Fillmore (1968) and others following him have used it. Syntactic case inflections are called 'gcase' for 'grammatical case' to distinguish them from semantic case. The GCASE attribute, with values for nominative (NOM), accusative (ACC), and genitive (GEN), is defined for pronouns in the lexicon. A composition rule also defines genitive determiners (see RULE DEF DET1 in Appendix A).

Figure IV-7 shows some of the syntax-oriented parts of the rule for prepositional phrase compositions. An application of PREPP1 to the instance "of the Lafayette" defines its CMU attribute as (COUNT); an application to "of seven thousand tons" defines its CMU attribute as (COUNT UNIT), since those are the unions of the values of the respective noun phrases. An application to "of surface displacement" assigns the value T (true) for the RELN attribute. We are not sure that the combination is a likely one in the task domain. Here as elsewhere we expect our factor scores to change as we collect and study more protocols. For the time being, "of surface displacement", "of draft", and the like are accepted as syntactically normal, although their form implies that we should perhaps assign the value MASS to the set of CMU values for relational nouns, so that "seven thousand tons of surface displacement" is treated syntactically as well as semantically in parallel with "seven thousand tons of water". (Compare also, "how much water?", ?"how much surface displacement?".)

When the head (NOMHEAD) of a nominal expression combines with a post-modifying prepositional phrase, a composition rule NH1 determines the CMU attribute of the resulting NOMHEAD by referencing the attribute of the prepositional phrase token. As a result, "surface displacement of the Lafayette" is (COUNT) in CMU value, while "surface displacement of seven thousand tons" is (COUNT UNIT) in CMU value. Figure IV-8 shows the relevant parts of the composition-rule definition.

Figure IV-7 Portions of the Composition-Rule Definition for Prepositional Phrases

```
RULE.DEF PREPP1     PREPP = PREP NP;
    ATTRIBUTES
        FOCUS,CMU,NBR,RELN,MOOD FROM NP;
    FACTORS
        GCASE = IF GCASE(NP) EQUAL "(NOM) THEN OUT ELSE OK;
    EXAMPLES
        OF THE LAFAYETTE (OK)
        OF 7000 TONS (OK)
        FOR WHICH SUB (OK)
        BY THE RUSSIANS (OK)
        OF THEY (OUT));
```

Figure IV-8 Portions of a Composition-Rule Definition for Nominal Expressions

```
RULE.DEF NH1     NOMHEAD = NOMHEAD PREPP;
    ATTRIBUTES
        CMU = IF RELN EQ T THEN
            GUNION(CMU(NOMHEAD),CMU(PREPP)) ELSE CMU(NOMHEAD),
        RELN,SUBCAT,NBR FROM NOMHEAD,
    FACTORS
        FSTWD = IF FSTWD(PREPP) EQ "OF THEN GOOD ELSE OK,
        MOOD = IF MOOD(PREPP) EQ "(WH) THEN POOR ELSE OK,
        RELN = IF RELN THEN VERYGOOD ELSE OK;
    EXAMPLES
        SPEED OF WHAT (POOR)
        SPEED OF TWENTY KNOTS (VERYGOOD));
```

A later composition rule propagates the values from NOMHEAD to NOM, so that the noun phrase composition-rule definitions with NOM constituents have access to them. Thus the differences in values for the two instances of NOM are referenced in the UNIT and RELN factors of NP11 to influence the choice of "a" as the alternative for "surface displacement of seven thousand tons" and "the" for "surface displacement of the Lafayette".

Notice, however, that NP4 cannot apply to "surface displacement of seven thousand tons" to combine it with a number expression, since the RELN factor blocks application to relational noun nominals, even those with UNIT as a CMU value. Given an utterance containing "which one's speed", an alternative parsing applying NP4 to "one -s- speed" to derive the phrase "one speed" will be quickly eliminated. So will a putative parsing of "five surface displacements of seven thousand tons".

4. Attributes and Factors in Higher Level Compositions

So far we have dealt mainly with noun phrases and prepositional phrases, showing how the attributes of their constituents are combined, propagated, and referenced in factors. One of these attributes, MOOD, is propagated to the highest levels, to sentences (S), and to the root category U. The question "What is the surface displacement of the Lafayette?" and its responses exemplify the two NP values DEC and WH that are also values for sentences and utterances. Sentences and utterances have other possible values. In "Is the surface displacement of the Lafayette seven thousand tons?" and "Does the Lafayette have a surface displacement of seven thousand tons?", the mood is yes-no (Y/N). In "List the surface displacements of the nukes", it is imperative (IMP). To state the matter more generally, the syntactic characteristics that distinguish the moods of sentences and utterances are the word class and mood of the initial constituent. If the initial constituent is a noun phrase, its mood is that of the sentence and is propagated to the utterance. When the initial constituent is an uninflected form of "be" or "do", the mood is yes-no; when it is an uninflected verb stem, the mood is imperative. For elliptical utterances, the mood may be undefined.

The noun phrases defined in the composition-rule definitions of Figure IV-6 derive their mood attributes from their first constituents. An article is always declarative, but number expressions ("five", "how many") and determiners ("this", "what") may be either DEC or WH. A determiner may also be the sole constituent of a noun phrase. The "what" of "What is the surface displacement?" is an example.

Our current vocabulary does not include verbs like "know" and "tell", which can embed WH questions like "Do you know what the surface displacement is?" For the time being, we assume that noninitial noun phrases are not likely to have the value WH. Echo questions, e.g., "You said what?" are not ruled out, but have lower scores. This is achieved by a WH factor in the determiner definition that lowers the score of an application when the acoustic pointer to the beginning of the word is not also at the beginning of the utterance. This constraint is in need of refinement, since we wish to recognize the kind of initial topic phrase that occurs frequently in our protocols, exemplified in "For what submarines is the surface displacement greater than seven thousand tons?" Additional refinements are necessary to recognize occurrences of "which" as a relative-clause introducer.

While permitting some noninitial WH noun phrases, factors in various composition-rule definitions lower the score for application still more for multiple occurrences of WH noun phrases within a phrase. "What is the speed of which submarine?" exemplifies an utterance whose score during parsing would be reduced by factors in the rules that apply to it.

For imperatives, the application of the composition-rule definition is blocked completely if the verb phrase contains a WH noun phrase. The relevant factor statement, labeled WH, appears in Figure IV-9. To see some of the possible effects of the WH factor, suppose that among the analyses considered as instances of an imperative are the following two alternatives:

List which submarines

and

List six submarines

The WH factor eliminates the first alternative.

For the WH factor to operate, the VP must have a mood attribute. In traditional linguistic analysis, the moods that are associated with verb phrases or verbs include the imperative mood but not the WH interrogative mood. In our definitions, verbs do not have imperative mood. However, sentences in which a verb phrase is the sole constituent may be imperative, if the instance of the verb phrase has the appropriate attributes. The remaining factor statements in Figure IV-9 specify what those must be. One of the attributes referenced there is called IMP. This is not a mood attribute, but an attribute of verb stems that marks whether they are potential heads of imperative sentences. Among the verb stems with the YES value for this attribute are "list" and "give"; "own" and "have" are marked NO. (We do not include the interpretation of "have" as "take", as in "have a piece of gum".)

Figure IV-9 Portions of the Composition-Rule Definition for Imperatives

```
RULE.DEF S4      S = VP;
    ATTRIBUTES
        FOCUS FROM VP,
        MOOD = "(IMP);

    FACTORS
        WH = IF MOOD(VP) EQUAL "(WH) THEN OUT ELSE OK,
        VOICE = SELECTQ VOICE(VP) WHEN PASS THEN OUT,
        IMP = SELECTQ IMP(VP) WHEN (YES, UNDEFINED) THEN OK
            ELSE OUT,
        NBR = IF NBR(VP) EQUAL "(SG) THEN OUT ELSE OK;

    EXAMPLES
        LIST WHICH SUBS (OUT)
        LIST SIX SUBS (OK)
        LISTS SIX SUBS (OUT)
        OWNED BY THE RUSSIANS (OUT)
        OWN THE SUBS (OUT);
```

The convergence of many attributes at the higher levels of composition makes possible many discriminatory judgments. Figure IV-10 shows something of the range of syntactically based judgments available at the sentence level.

Figure IV-10 Portions of a Composition-Rule Definition for a Sentence Composition

```
RULE.DEF S3     S = NP:NP1 AUXB NP:NP2;
    ATTRIBUTES
        MOOD,FOCUS,CMU,RELN  FROM NP1,
        AFFNEG FROM AUXB;
    FACTORS
        NBRAGR1 = IF  CMU EQUAL "(UNIT) THEN
             [IF NBR(AUXB)EQUAL "(SG)THEN OK ELSE OUT]ELSE
              IF GINTERSECT(NBR(NP1),NBR(AUXB)THEN OK ELSE OUT,
        NBRAGR2 = IF  CMU(NP2) EQUAL "(UNIT)  THEN OK, ELSE
                IF GINTERSECT(NBR(NP2),NBR(AUXB))THEN OK ELSE OUT,
        FOCUS = IF FOCUS(NP1) EQ "INDEF AND FOCUS(NP2) EQ "DEF
          THEN POOR ELSE OK,
        GCASE1 = IF GCASE(NP1) EQUAL "(ACC) THEN OUT ELSE OK,
        GCASE2 = IF GCASE(NP2) EQUAL "(ACC) THEN OUT ELSE OK,
        MOOD1 = IF MOOD EQUAL "(WH) THEN GOOD ELSE OK,
        MOOD2 = IF MOOD EQUAL "(WH) AND MOOD(NP2) EQUAL "(WH)
             THEN POOR ELSE OK,
        AFFNEG = IF MOOD EQUAL "(WH) AND AFFNEG EQ "NEG THEN BAD
           ELSE OK,
        RELN = IF RELN EQ "T THEN IF CMU(NP2) EQUAL "(UNIT)
                THEN VERYGOOD ELSE OK,
        PERSAGR = IF GINTERSECT(PERS(NP1),PERS(AUXB))
                 THEN OK ELSE OUT;
    EXAMPLES
        THE LAFAYETTE IS A SUBMARINE (OK)
        THE LAFAYETTE IS SUBMARINES (OUT)
        A LAFAYETTE IS THE SUBMARINE (POOR)
        THEM ARE SUBMARINES (OUT)
        WHAT IS THEM (OUT)
        WHAT IS IT (GOOD)
        HOW MANY ARE WHAT (POOR)
        IT AM A LAFAYETTE (OUT)
        WHAT ISN'T THE SURFACE DISPLACEMENT OF THE LAFAYETTE (BAD)
        WHAT IS THE SURFACE DISPLACEMENT (GOOD)
        THE SURFACE DISPLACEMENT IS 7000 TONS (VERYGOOD);
```

The S defined by the composition-rule definition for S3 has the mood, focus, CMU, and relational noun attributes of its subject noun phrase. An attribute AFFNEG is copied from the AUXB constituent. If that constituent contains a "not" or "-n't", the value is NEG; otherwise it is AFF.

In the factor statements, the PERSAGR factor tests for agreement between the so-called pronouns (including the indexical forms for speaker and hearer) and the auxiliary constituent. The two grammatical case factors, GCASE1 and GCASE2, require that the grammatical cases of the two NPs are not accusative. These traditional syntactic agreement tests block application of the composition rule to putative expressions like "it are" and "they is". "Them is" is doubly blocked.

Such traditional tests can be further elaborated and refined to reduce alternatives and make predictions. We are constantly finding new opportunities to introduce new constraints as we proceed. For example, it is not very likely that the noun phrases of S3 will be genitives, although genitive determiners for noun phrases are not uncommon. If one of them is genitive, it seems more likely that it will be the second. For example, after asking "What is the draft of the George Washington?", the next question may be "What is the Lafayette's?" (or just "The Lafayette's?"). For both noun phrases to be genitive seems least likely of all. Of course, it is always possible to construct acceptable examples for special contexts.

Factor statements referencing the genitive case attribute value are currently being elaborated for the next revision of the composition rules. Constraints on genitives are especially desirable because the genitive suffix, the plural suffix, the singular verb suffix, and the reduced form of "is" are all homophonous, giving rise to many ambiguities. "The Lafayette -s" could be genitive, plural, genitive plural, a reduced "Lafayette is", "Lafayette's is", or even part of "the Lafayette submarine class".

Some of the remaining factor statements in Figure IV-10 are less traditional and more performance oriented than the person and case agreement factors. The first MOOD factor enhances the score if the composition-rule definition is applied to an instance of a WH question. This partial judgment is based on the very high frequency of WH questions formed on this pattern in our protocols. Answers were typically elliptical noun phrases or nominals, so that the declarative form of S3 was rare. We may wish to reset this factor dynamically for discourse in the computer consultant task domain. If the consultant has just asked the user "What is that tool you are using?", we would predict a declarative utterance from the user, possibly on the same pattern, "This is a socket wrench." Resetting the MOOD1 factor to enhance the score if the mood is declarative would lower the score, relatively, for the phonetically similar but inappropriate "Which is a socket wrench."

The second mood factor, MOOD2, lowers the score if both noun phrases are WH. The AFFNEG factor references both the MOOD and AFFNEG attributes and reduces the score greatly (BAD) if the instance is purportedly a negative WH question, for example, "What isn't the surface displacement?" Genuine requests for negative information occur in highly circumscribed situations. The rhetorical question is not a genuine request for information (e.g., "Who wouldn't like to be rich and famous!"). "Who isn't here?" is ordinary only if there is an established and limited list of people who are expected to be present, as in a classroom. "What isn't your name?" and "Where don't you live?" are patently absurd.

The constraint imposed on occurrences of WH interrogatives that are also negative is stated in syntactic terms, but it is essentially due to pragmatic forces as well as semantic ones. Very similar forces appear to be at work in observed tendencies for the first NP in the composition defined by S3 to be indefinite in focus only when the second one is also. Stated oversimply, in coherent discourse, the things already talked about--the 'old' information--tends to come first. What is predicated about it--the 'new' information--tends to follow. Old information is information that has already been talked about and established in the discourse, so that it is likely to be encoded in definite noun phrases. These are likely to be in subject position, so that the sentence they introduce is consistent with preceding sentences. New information tends to be introduced in indefinite noun phrases. The next mention of the 'same thing' will then be old information, eligible for definite focus. Consequently, "A Lafayette is that submarine." seems peculiar, relative to "That submarine is a Lafayette." "A Lafayette is it." is still more peculiar. It is not odd for both NPs to be indefinite, as in "Lafayettes are submarines." Definitions and generic statements often follow this pattern. In statements used to identify an object that is, in some sense, 'in view', it is common for both NPs to be definite, as in "That is the one." Only the pattern of indefinite followed by definite noun phrase seems unusual and inverted. (A classic example is "A wise bird is the owl.") This discourse-based probabilistic tendency is expressed in the FOCUS factor of S3.

The remaining factor statements appeal to attributes that have been propagated from the noun stems in the lexicon. The relational noun factor (RELN) enhances the score for applications to instances in which the subject noun phrase is a relational noun and the predicate noun phrase is a unit expression, as in "The surface displacement is seven thousand tons.", and "The width is two inches."

The RELN factor is another example of a performance-based characteristic that is not traditionally considered to be within the province of syntax. On the other hand, matters like number agreement have always been central to syntax. It is particularly interesting, therefore, that the number agreement constraints for S3 cannot be properly stated without appealing to a distinction established for noun stems on the basis of their semantic functions. To state number agreement constraints, noun stems denoting units must be marked separately. Sentences like "These are a submarine.", "These is a torpedo tube.", "These is missile launchers.", and "This are subs." are clearly ungrammatical, and the ungrammaticality is clearly due to the fact that one of the constituents differs in grammatical number from the other two. However, "The surface displacement is seven thousand tons," is wholly grammatical in spite of the fact that two of the constituents are singular and the third is plural. Its inverted form, "Seven thousand tons is the surface displacement.", is also wholly grammatical with respect to number agreement, although the FOCUS factor will reduce its score because of the inversion.

5. Syntax and Prosodics

The previous discussion has centered around demonstrations of semantic features of words and phrases that are referenced by syntax-oriented factor statements. Acoustic-phonetic features of pitch, stress, duration, and pause can also be referenced by syntax-oriented factors to assess the scores of phrases on the basis of their prosodies. For example, the composition rules for yes/no questions like "Is the surface displacement more than seven thousand tons?" and "Does it have torpedo tubes?" have two prosodic factors, exemplified in Figure IV-11.

The pitch attribute is to be provided by the acoustic component, which checks direction of the pitch at the beginning and end of the sentence. The pitch factor enhances the score if the pitch is rising. (The condition "IF VIRTUAL THEN OK" does not concern us here. It enables the parser to ignore a factor when nodes are constructed predictively for testing.)

Figure IV-11 Prosodic Elements in a Composition Rule

```
RULE.DEF S8     S = AUXB NP;NP1 NP;NP2;
    ATTRIBUTES
        PITCHC = FINDPITCHC(PLEFT,PRIGHT);
    FACTORS
        STRESS = IF VIRTUAL THEN OK ELSE
            SELECTQ STRESS(AUXB) WHEN UNREDUCED THEN GOOD,
        PITCHC = IF VIRTUAL THEN OK ELSE
            IF PITCHC EQ "HIRISE THEN GOOD ELSE OK;
    EXAMPLES
        IS IT A LAFAYETTE?? (GOOD) --
            ?? INDICATES A HIGH RISING PITCH
        'S IT A LAFAYETTE (POOR);
```

The pitch attribute and factor have not been tested and are admittedly imprecise. As currently written, they are essentially place holders for more accurate and testable statements. We know, for instance, that the total pitch contour of a sentence is not relevant for signaling its syntactic type, but only the contour of the last tone group. We need to be able to find the last tonic stress and determine the direction of pitch from that point to the end.

The stress factor, which also has not been tested, enhances the score if the auxiliary contains a full vowel. Utterance-initial auxiliaries are more resistant to vowel reduction than medial ones, so that while "It's a Lafayette.", with elision of the vowel is likely, "'S it a Lafayette." is unlikely, though possible.

There are also stress factors on rules combining auxiliaries with negatives. For instance, if the negative is the reduced form "-n't", then the auxiliary should contain a full vowel. Thus while "it is not", "it's not", and "it isn't" are to be expected, "it'sn't" is not (see RULE.DEF B3 in Appendix A.)

Pitch and stress pose many well known problems that have deterred us from further development of attributes and factors based on them. One problem is the scarcity of data on their use in spontaneous speech. SRI, SDC, and SCRL have agreed on a set of conventions for transcribing protocols from our task domains, marking pauses (both silent and 'filled'), tonic syllables, and pitch direction. Transcriptions supplied by SCRL of selected parts of protocols will give us some of the much-needed data and lead to adjustments in the statements concerning pitch and stress that are now only place holders in the Language Definition.

Another problem, however, is the complexity of the relationship between intonation and syntax. The notion that people consistently end statements with falling pitch and questions with rising ones is clearly erroneous. It is more accurate to say that nonfalling pitch signals incompleteness, but this statement is rather vague. While yes/no questions appear more likely than other sentence types to end with rising pitch, even they tend to exhibit falling pitch if '' alternatives are specified as in "Does it or doesn't it?" In addition, the acoustic correlates of both pitch and stress are not well known and the nature of their combination in intonation patterns is a continuing puzzle in both linguistic and acoustic research.

These problems have led us to turn our own efforts, in conjunction with SDC and SCRL, toward developing attribute and factor statements based on pause and duration. We believe that these two features are relatively easier to define and measure.

In any case, it is necessary to determine the presence and duration of pauses within an utterance and to separate the silences that are manifestations of voiceless stops from those that are unfilled pauses marking syntactic boundaries or hesitations. Pauses are obviously useful in determining word boundaries, since even nonfluent speakers rarely pause within a word. If we cannot assume that every word is followed by a pause, we may assume that any pause occurs at the end of a word. Incidentally, use of pause for word boundary determination will entail some changes in our current treatment of items like "submerged displacement" as unanalyzed units. However, submarine names like "Ethan Allen" and "George Washington" meet the criterion for occurring without internal pause even for nonfluent speakers.

We are also attempting to establish attributes of duration for words and phrases, with values LONG and SHORT, so that pause factors can reference them and enhance the scores for instances in which pauses appear in the appropriate place between long phrases. For example, in long number phrases, a pause is acceptable between the largest number category in the phrase and the following additive phrase, as in "fifty thousand pause five hundred and sixty-two", and is inappropriate elsewhere, as in "fifty thousand five pause hundred and sixty two". (See RULE.DEF NUMS and RULE.DEF NUM 16 in the Appendix.)

To investigate the acoustic parameters of these prosodies as they appear in spontaneous speech, colleagues at SDC undertook to handmark one of the submarine protocols for durations of pauses and of words. The SDC concordance program, KWICO, brought together all occurrences of each word and pause, arranged in order of increasing duration. Figure IV-12 is a sample from the concordance. The arrangement permits us to make comparisons and form hypotheses regarding the distribution of pauses and their correlation with word and phrase boundaries and with word durations.

We expect to use word durations in characterizing a category of 'function' words, whose members are typically short and unstressed, for example, articles and most prepositions. The data, so far, indicate that these words have shorter durations than content words, which have at least one unreducible vowel. There is significant lengthening, however, before pause, so that the durations of function words and content words overlap. For example, while the typical duration of "of" was much shorter than that of any content word, the longest duration was 32 segments (320 milliseconds), which is not uncommon for many monosyllables.

```
                                                                                                    PAGE 007

             LIST027 ALL033 CLASSES064 OF032 PAUSE016 OF020 NUCLEAR046 BALLISTIC057 MISSILE045 PAUSE040 UNITED029 STATES032  003001
                      PAUSE006 AND039 THE010 LENGTH059 OF021 THE010 GEORGE030 WASHINGTON058 PAUSE025                          003012
 7 IS014 THE037 PAUSE068 THE025 LENGTH045 PAUSE020 CF021 THE013 ETHAN030 ALLEN039                                             C03011
                        LIST027 ALL033 CLASSES064 OF032 PAUSE016 OF020 NUCLEAR046 BALLISTIC057 MISSILE045 PAUSE040 UNITE     003001
                                          SUBSET064 ON020 ALL034 PAUSE045 UNITED042 STATES036 SUBMARINES080 PAUSE097 AHC24   003064
                        HOW010 MANY017 DIESEL029 PATROL039 SUBMARINES047 DO010 THE004 BRITISH035 HAVE019                      003034
                          HOW016 MANY020 DIESEL042 PATROL044 SUBMARINES041 DO012 THE005 RUSSIANS042 HAVE024                   003035
 66 AND023 HOW017 MANY042 CLICK012 AHC18 DIESEL033 PATROL051 SUBMARINES042 DO012 THEY015 HAVE039                              003032
             HOW035 HOW020 MANY031 PAUSE071 DIESEL070 PAUSE003 GUIDED045 MISSILE045 PAUSE101 SUBMARINES092 AND020 HOW018 MAN   003025
 THOSE029 PAUSE010 WHAT012 IS015 THE010 MAXIMUM072 PAUSE004 AND035 PAUSE021 MINIMUM052 PAUSE087 COMPLEMENT048 PAUSE005         003068
                 AND038 THE007 UNITED035 KINGDOM038 PAUSE005                                                                  003021
               COMPLETER'S039 BEING021 SLOW029 TODAY025 PAUSE005                                                              003027
 AND035 PAUSE021 MINIMUM052 PAUSE087 COMPLEMENT048 PAUSE005                                                                   003068
                                                    PAUSE005 AH041 HOW022 MANY052 PAUSE146 DIESEL054 PAUSE007 TRAINING042     003029
                                                    PAUSE005 HOW019 MANY021 TORPEDO063 TUBES060 PAUSE054 AH026 DOES032 THE   003013
 SUBMARINE043 HAS024 AC07 COMPLEMENT043 CF005 AC08 PAUSE005 HUNDRED025 AND006 PAUSE008 FIVE025                                C03069
 08 BRITISH064 PAUSE020 HAVE016 AH014 OAH015 AH038 PAUSE006                                                                   003033
                     AND019 THE005 RESOLUTION084 PAUSE006                                                                     003051
                                                    PAUSE006 AND031 HOW025 MANY042 PAUSE091 NUCLEAR069 PAUSE034 FLEET033 S   003024
                                                    PAUSE006 AND039 THE010 LENGTH059 CF021 THE010 GEORGE030 WASHINGTON058    003012
 032 SUBMARINES068 PAUSE047 WITH038 PAUSE090 AH027 PAUSE006 BEAM038 LESS043 PAUSE007 THAN023 THIRTY037 FEET038                 003062
                                                    PAUSE006 CF014 THE007 GEORGE033 WASHINGTON053                            003006
 52 TUBES037 FOR007 THE009 GEORGE035 WASHINGTON054 PAUSE007                                                                   003057
 7 WITH038 PAUSE090 AH027 PAUSE006 BEAM038 LESS043 PAUSE007 THAN023 THIRTY037 FEET038                                         003062
  PAUSE005 AH041 HOW022 MANY052 PAUSE146 DIESEL054 PAUSE007 TRAINING042 SUBMARINES062 DOES019 THE005 UNITED030 STATES031       003029
                  REPEAT032 THE007 LIST030 AGAIN038 PAUSE008                                                                  C03067
 MPLEMENT043 OF005 AC08 PAUSE005 HUNDRED025 AND006 PAUSE008 FIVE025                                                           C03069
                                                    PAUSE008 HOW030 MANY044 PAUSE029 TORPEDO045 TUBES030 DOES011 THE009 GF   003055
 1 SUBMARINE062 PAUSE031 HAS019 THE018 GREATEST056 PAUSE008 NUMBER052 OF036 MISSILE039 LAUNCHERS046                           003038
 HAT020 PAUSE076 HOW016 MANY042 PAUSE064 SUBMAR040 PAUSE008 UNITED037 STATES018 SUBMARINES053 PAUSE025 HAVE033 PAUSE015 L    003063
                PAUSE041 THANK027 YOU008 COMPUTER045 PAUSE009                                                                 003061
                                                    PAUSE009 HOW027 MANY031 PAUSE020 DIESEL053 GUIDED037 MISSILE027 SUBMAR    003028
                   FOR004 THE010 SOVIET049 UNION031 PAUSE010                                                                  003020
                     PAUSE034 HOW013 MANY028 AH054 PAUSE010 NUCLEAR060 ATTACK041 SUBMARINES057 DOES021 THE006 UNITED028 S   C03022
                                    OF006 THOSE029 PAUSE010 WHAT012 IS015 THE010 MAXIMUM072 PAUSE004 AND035 PAUSE021 MINI     003068
  UNITED042 STATES036 SUBMARINES080 PAUSE097 AH024 PAUSE010 WITH031 PAUSE044 BEAM030 LESS025 THAN017 THIRTY032 FEET048        003064
                                                    PAUSE011 HOW013 MANY013 TORPEDO040 TUBES029 DOES013 THE010 ETHAN035 AL    003054
 E033 PAUSE015 LENGTH026 LESS024 THAN013 THIRTY027 PAUSE013 FEET016                                                           003063
                              WHAT013 IS019 THE039 PAUSE013 LENGTH034 OF011 THE005 LAFAYETTE057                               C03004
  WHICH033 SUBMARINES049 HAS031 THE013 GREATEST050 PAUSE013 NUMBER036 OF007 MISSILE038 LAUNCHERS047                           003037
  PAUSE074 NUMBER031 OF004 MISSILE038 LAUNCHERS056 PAUSE013 THE012 SUBMERGED051 SPEED052 PAUSE044 AND026 THE009 NUMBER037     003057
                                 PAUSE014 AH022 PAUSE013 THE015 GEORGE032 WASHINGTON052                                       00305C
                                                    PAUSE014 AH022 PAUSE013 THE015 GEORGE032 WASHINGTON052                   003050
                                                    PAUSE014 ALL009 RIGHT028                                                 003036
   AND075 THE054 PAUSE020 GEORGE037 WASHINGTON049 PAUSE015                                                                    C0301C
                                                    PAUSE015 HOW016 MANY028 NUCLEAR054 CRUISE041 MISSILE032 SUBMARINES066    003023
 NITED037 STATES018 SUBMARINES053 PAUSE025 HAVE033 PAUSE015 LENGTH026 LESS024 THAN013 THIRTY027 PAUSE013 FEET0..              003063
 HO32 PAUSE040 NUMBER029 OF009 TORPEDO051 TUBES048 PAUSE015 NUMBER032 OF006 REACTORS065 PAUSE071 NUMBER033 OF010 MISSILEG    003002
                                           WHICH025 PAUSE015 WHICH015 SUBMARINE043 HAS024 AC07 COMPLEMENT043 OF005 AC08 PA    003069
                 LIST027 ALL033 CLASSES064 OF032 PAUSE016 OF020 NUCLEAR046 BALLISTIC057 MISSILE045 PAUSE040 UNITED029 S       0030C1
                                                    PAUSE018 THE024 SUBMERGED034 SPEED021 OF003 THE010 ETHAN023 ALLEN036      003059
                                                    PAUSE019 FOUR027 EACH036                                                 003026
                   PAUSE009 HOW027 MANY031 PAUSE020 DIESEL053 GUIDED037 MISSILE027 SUBMARINES060 DOES023 THE006 U              003028
                               AND075 THE054 PAUSE020 GEORGE037 WASHINGTON049 PAUSE015                                       003010
 FLC36 SUBMARINES080 AH033 DO015 THE008 BRITISH064 PAUSE020 HAVE016 AH014 OAH015 AH038 PAUSE006                                003033
```

FIGURE IV-12 SAMPLE CONCORDANCE PAGE FOR PROSODIC DATA

This longest instance occurred before pause. The next longest instance was only 21 segments and was not followed by pause.

In general, the data bear out some observations reported in the literature to the effect that while words can be lengthened with considerable freedom, they cannot be shortened below a certain limit if they contain stressed syllables. There appears to be an inherent lower limit to the duration of content words. These durational attributes should be helpful in reducing the search space for words to fill gaps in partially analyzed utterances.

6. Lexical Subsetting

One method for reducing the search space for word mapping was introduced into the parser in the fall of 1974. The method is referred to as 'lexical subsetting'. Early this year, a preliminary scheme for classifying words was arrived at cooperatively with SDC, and SDC undertook to implement the necessary acoustic routines. The classifications are changing, so that the examples given below are no longer representative. They are given only to illustrate the concept of lexical subsetting concretely.

The general idea of lexical subsetting is that when the acoustic component is called with a pointer to a position in the utterance, it returns a subset of the vocabulary; that is, a list of candidate words that can begin there, if processing is from the left to the right, or end there, if it is from the right to the left. The candidates are those whose first syllable (when subsetting to the right) or last syllable (when subsetting to the left) are broadly correlatable with the acoustic parameters for the neighborhood. For example, if processing is to the right and the nearest steady state looks like a high front vowel, possible candidates include "speed", "Ethan.Allen", "diesel", "Seawolf", and "three", but not "Lafayette" or "submarine" or "four". If processing is to the left, candidates include "speed", "three", and "submarine", but not "Ethan.Allen" or "Seawolf" or "four".

Our first classification for lexical subsetting appears in Figure IV-13. Words are classified according to type of vowel of the first and last syllables (FV and LV) and according to sibilant or nonsibilant first and last consonants (FCSIB and LCSIB). A word may belong to more than one FV or LV class if it has forms that differ with respect to vowel types in the first or last syllable. For example, "eleven" belongs to FV1 because it is sometimes pronounced /IY:1 LEH:2 VAXN:0/ and also to FVMISC because it is sometimes pronounced /AX:0 LEH:2 VAXN:0/. Unstressed function words are not classified, since they are always predicted before they are looked for.

It should be possible to return unions and intersections of these classes. For example, if the word actually present is "speed", the lexical subsetting routine should be able to return a list of forms from the intersection of FV1 and FCSIB when

Figure IV-13 A Classification for Lexical Subsetting

FV1: ETHAN.ALLEN SEAWOLF SPEED DIESEL THREE
WE FEET ELEVEN WHICH FIF SIX LITTLE
GREAT EIGHT EITHER NEITHER FIVE NINE

LV1: GUPPY.THREE SUBMARINE SUBMERGED.SPEED WE
SURFACE.SPEED SPEED THREE FEET WHICH MANY
FIF GREAT EIGHT FIVE NINE LAFAYETTE

FV2: SURFACE.SPEED SURFACE.DISPLACEMENT THIR
FOUR TORPEDO.TUBE MORE ARE SHORT

LV2: THIR FOUR MORE ARE SHORT EITHER NEITHER

FV3: DO HOW TWO FEW U.S NUC WHO
DONT OWN BOTH

LV3: DO HOW TWO TORPEDO.TUBE FEW DONT OWN
WHO BOTH

FV4: GUPPY.THREE SUBMERGED.SPEED
SUBMERGED.DISPLACEMENT SUBMARINE DRAFT WHAT MUCH
ONE HUNDRED NONE SOME TEN TWEN LENGTH TWELVE
ALL HAVE THAN THAT NINE FIVE

LV4: DRAFT WHAT ONE NONE SOME LAFAYETTE
ALL TEN TWEN LENGTH TWELVE HAVE THAN THAT MUCH
NINE FIVE

FCSIB: SUBMERGED.SPEED SURFACE.SPEED SPEED SEAWOLF
SUBMARINE SIX SUBMERGED.DISPLACEMENT SURFACE.DISPLACEMENT
SOME SHORT

LCSIB: U.S HAS SIX

FVMISC: ETHAN.ALLEN LAFAYETTE ELEVEN
WHICH FIF LITTLE MANY HOW WHAT HUNDRED
NONE TEN TWEN TWELVE LENGTH HAVE THAN THAT
ALL

LVMISC: ETHAN.ALLEN SEAWOLF ELEVEN WHICH DIESEL
SUBMERGED.DISPLACEMENT SURFACE.DISPLACEMENT FIF LITTLE
MANY HOW WHAT HUNDRED NONE TEN TWEN TWELVE
LENGTH HAVE THAN THAT ALL EITHER NEITHER

subsetting on the right. Presumably a union should be returned if the acoustic parameters correlate with a vowel quality falling within the ranges of two different vowel types. However, we hope to establish syllable nucleus types in which there is minimal overlap.

A new classification will be ready for testing shortly. Simulated tests on text with and without subsetting lead us to expect very substantial improvement in efficiency with even a crude subsetting scheme. As the vocabulary increases, we will need the reduction in search space even more than we do now. The bulk of a vocabulary is generally in the set of noun stems. Many of these will be semantically and syntactically so much alike that only acoustics affords a basis for distinguishing among them. For example, if 100 entries are names of ships, there will be 100 alternatives for filling a gap in an utterance if the only constraint is that it be filled by a ship's name. The phonological characteristics of the first and last syllables, together with prosodic features of stress and duration, are the attributes that appear to be most serviceable in such circumstances.

V SEMANTICS

Prepared by Gary H. Hendrix

Contents:

A. Introduction

Since our work this year began in the context of a joint system development effort with SDC and the introduction of a new kind of task domain, we had an unusual opportunity to reflect on the approach to semantics we had been following and to consider possible alternatives and improvements. After careful consideration, a semantic system based on semantic networks was implemented and tested. While this module continues to supply semantic support for the total speech understanding system, the experiences and insights gained from our programming efforts have suggested so many improvements that we are now making substantial revisions to take advantage of our better understanding of network structures and procedures for partitioning them. Details concerning our selection of the new semantic representation, the theory behind partitioned semantic networks, our current implementation, and plans for improvements are discussed in this section.

In our previous system, the semantics relied heavily on a process model, which formalized the steps entailed in the maintenance and repair of a particular kind of small appliance. While we continue to believe that approach is viable for the semantics of dynamic domains, like the computer consultant task, the data management task domain, assumed through our cooperation with SDC, focuses on the retrieval of static facts from a data base and deemphasizes the notion of process. Since process models are singularly unsuitable for the representation of static data bases, we have considered alternative representation schemes in the hope of finding a comprehensive system capable of accommodating both task domains and, hopefully, any others that we might select in the future. In particular, we have considered relational tables (following Codd, 1970) and semantic networks (as used by Simmons, 1973; Shapiro, 1971; Rumelhart and Norman, 1973, and Schank, 1973). The tabular scheme offers the advantage of compact representation for data bases (such as attributes of submarines or the information on ships for the Mediterranean scenario), which are regular, tabular, and dense. Semantic networks, however, offer such recognized advantages as a convenient bidirectional linkage between semantically related data items and an inherent facility for associating deep conceptual case systems with event types. Furthermore, system inputs may be translated into network notations paralleling the notations used for the encoding of data base knowledge.

The representation we have adopted is a variation on conventional networks that allows quantification and categorization to be handled easily and that facilitates both the translation from English into network representations and the building of constructs for discourse analysis. This variation also facilitates the encoding of process models in networks and will eventually allow us to tap our previous work in process modeling to extend our system to dynamic domains. We have extended the encoding power and flexibility of semantic networks by introducing an augmentation in which the nodes and arcs of networks are partitioned into 'spaces'.[1] These spaces allow knowledge to be bundled into units which help to condense and organize the data base. Since many of the distinctive aspects of partitioned networks may be presented more easily in terms of the computer consultant task domain than in terms of the data management task domain, our initial examples are taken primarily from the former source.

------
[1] A short introduction to partitioned semantic networks is contained in Hendrix (1975).

B. Theoretical Basis of Partitioned Semantic Networks

1. Basic Network Notions

In its simplest form, a semantic network is a set of nodes interconnected by an accompanying set of arcs. A node may be used to represent an 'object', where an object may be virtually anything, including physical objects, relationships, sets, events, rules, and utterances. Arcs are used to represent certain 'primitive' omnichronic (i.e., time invariant) relationships, although such relationships may also be represented as nodes.

A feeling for how nodes and arcs are organized to represent various facts may be gained by considering the network of Figure V-1. In this network the node 'PHYSICAL.OBJECTS' (single quotes are used to designate nodes) represents the set PHYSICAL.OBJECTS, the set of all physical objects. Likewise, node 'MACHINE.PARTS' represents the set of all machine parts. The arc labeled "s" from 'MACHINE.PARTS' to 'PHYSICAL.OBJECTS' indicates that MACHINE.PARTS is a subset of PHYSICAL.OBJECTS. Similarly, the network indicates that BOLTS is a subset of MACHINE.PARTS and that B, an element of BOLTS (connected by an arc labeled "e"), is a particular bolt. Following the hierarchy of another family, [illegible] is a particular box, an element of BOXES, which is a subset of CONTAINERS, which is a subset of PHYSICAL.OBJECTS.

Node 'C' encodes a containing situation, an element of the situations set <sit-contain>, a subset of SITUATIONS, which is the set of all situations. In particular, 'C' represents the containing of bolt B by box X from time T1 until time T2. The various component parts of situation C are associated with it through special deep case relationships. For example, in the network there is an arc labeled "content" from 'C' to 'B'. This arc indicates that B is the #@content of situation C, where the notation "#@content of C" means "the value (#) of the content attribute (@) of C." Similarly, X is the #@container of C while T1 and T2 are the #@start-time and #@end-time, respectively.

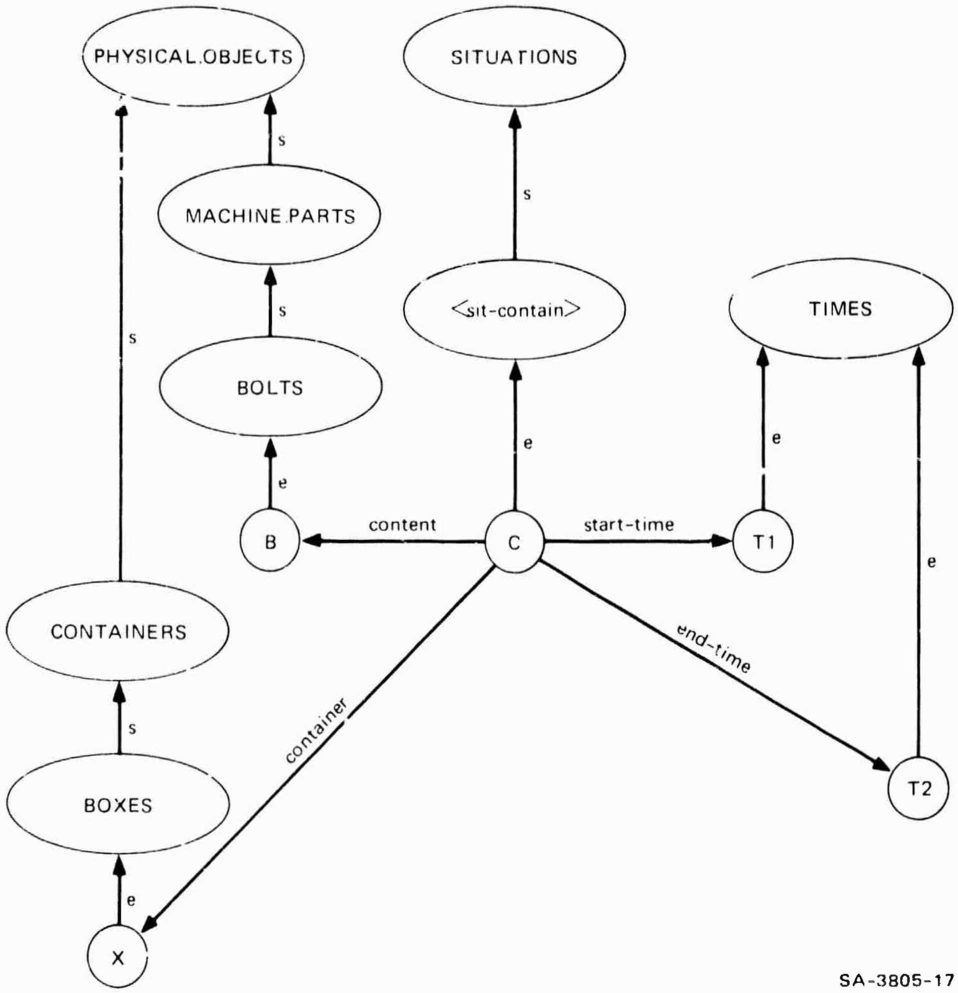


FIGURE V-1 A TYPICAL NET FRAGMENT

As a general principle, arcs encode only element, subset, and case relationships. (Under one interpretation, element and subset relations may be viewed as deep cases also.) Arcs are never allowed to encode relationships, such as ownership, which are time bounded.

2. Net Partitioning

The central idea of net partitioning is to separate the various nodes and arcs of a network into units called spaces. Every node and every arc of the overall network is assigned to exactly one space, with all nodes and arcs that lie in the same space being distinguishable from those of other spaces. While nodes and arcs of different spaces may be linked, the linkage must pass through certain boundaries that separate one net space from another.

Net spaces are typically used to delimit the scopes of quantified variables and to distinguish alternative hypotheses (during parsing and planning). However, before taking up such practical applications, consider the simpler (if atypical) network partitioning exhibited in Figure V-2. As shown, each space of the partitioning is enclosed within a dotted line. For example, space S1 is at the top of the figure and includes nodes 'PHYSICAL.OBJECTS', 'BOLTS', '<sit-contain>' and 'BOXES'. S1 also includes the two s arcs indicating that the set of BOLTS and the set of BOXES are subsets of the set of PHYSICAL.OBJECTS. In our

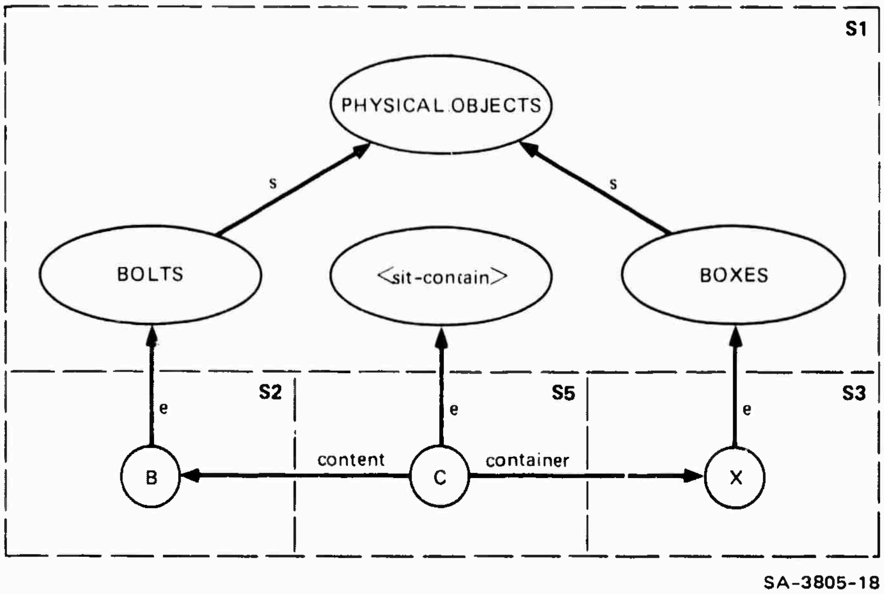


FIGURE V-2 A SAMPLE NET SPACE PARTITION

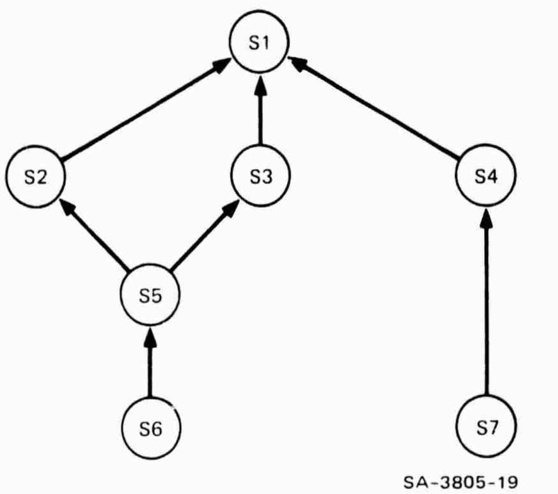


FIGURE V-3 A NET-SPACE PARTIAL ORDERING

diagramatic representations of semantic nets, an arc belongs to a space iff the arc's label is written within the dotted line boundaries of the space. Thus the e arc from 'B' to 'BOLTS' lies in space S2.

The various spaces of a partitioning are organized into a partial ordering such as that shown in Figure V-3. In viewing the semantic network from some point S in this ordering, only the nodes and arcs are visible that lie in S or in a space above S in the ordering. Thus, for example, from space S2 of Figures V-2 and V-3, only the nodes and arcs lying in S2 or S1 are visible. In particular, it is possible to see that B is a BOLT and that BOLTS are PHYSICAL.OBJECTS, but it is not possible to see that X is a BOX. From space S5, information in spaces S5, S3, S2, and S1 is visible. Hence, from S5, the whole of the semantic network of Figure V-2 may be seen.[2]

In practice, partitioned networks are constructed by creating empty net spaces, adding them to the partition ordering, and then creating nodes and arcs within each newly created space. The use of partitioning in the encoding of quantified statements and categories is the subject of the next two sections.

------

[2] For certain applications, the net may be inspected one space at a time. For example, it is possible to query the net in such a way that only nodes and arcs lying in space S2 are visible even though information in S1 is normally visible whenever S2 is inspected.

3. Quantified Statements

In addition to an ability to encode specific facts (such as the containing event encoded in Figure V-1), a semantic system needs some facility for grouping sets of similar facts into units and for allowing these facts to be represented collectively through some sharing mechanism and to be conceptualized as an integrated whole. An ability to encode generalized information (in the form of quantified expressions) is of considerable importance since it is often impractical (or even impossible) to record the same information by a collection of individual specific statements both because of the very number (possibly infinite) of statements required and because details of particular individuals may not be explicitly known. Furthermore, since quantification is a component of language, an ability to encode quantifiers is vital to the understanding of certain classes of English sentences (e.g., "Are all subs in the Russian fleet nuclear powered?", "Do some U.S. boats have more than five torpedo tubes?")

As an example of how quantification is handled in partitioned networks, consider the network of Figure V-4 which encodes the statement

> Every bolt in the box is 3/4 inch long and has a nut screwed onto it.

In this network, the node 'GS' represents the set of all general statements (the set of statements involving universal quantifiers or, under another interpretation, the set of recurring patterns of events). The node 'g' represents the particular statement (set of events) cited above.

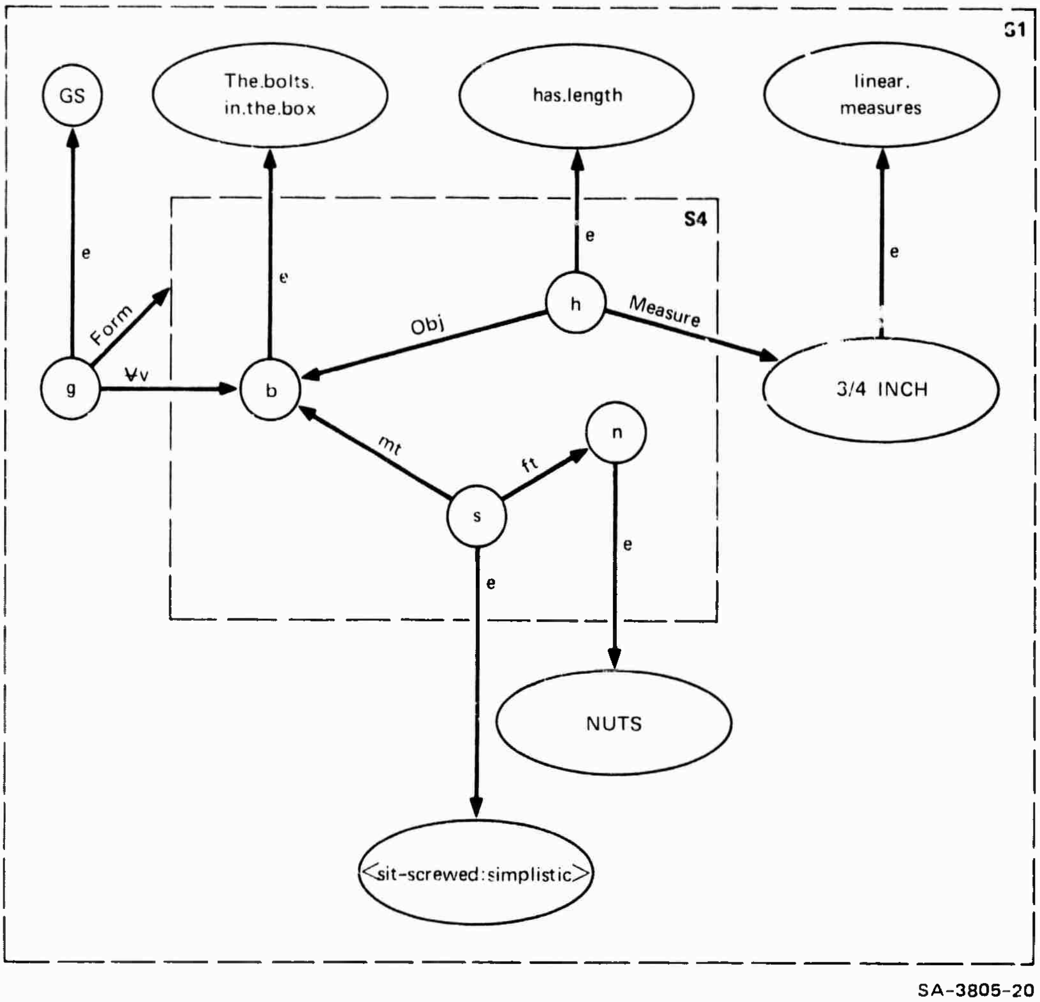


FIGURE V-4 EVERY BOLT IN THE BOX IS 3/4-INCH LONG AND HAS A NUT SCREWED ONTO IT

Characteristically, a general statement encodes a collection of separate circumstances all of which follow the same basic pattern. This basic pattern is represented by the #@form of the general statement. The #@form of g is encoded by a net space, S4, which lies just below S1 in the partition ordering. (When one net space is pictured inside another, the inner space is below the outer in the partition ordering.) This net space may be thought of as a super-node containing its own internal structure and representing a composite variable which takes on a different value for each of the instantiations of the recurring pattern. Each node and arc within the space of the super-node may be thought of as a subvariable.

General statements are also typically associated with one or more universally quantified variables which are allowed to assume values from some specified range. Statement g, for example, has a universally quantified variable b given by the value of its @Vv attribute. Note that variable b is necessarily a part of the #@form of g (i.e., 'b' lies in space S4). From node 'b' there is an e arc to the set the.bolts.in.the.box, indicating that the value of b (written #b) must be taken from the range set the.bolts.in.the.box. The node 'the.bolts.in.the.box' has been created especially to help encode the general statement. Its meaning may be inferred subsequently when the.bolts.in.box.X is defined by the network of Figure V-5.

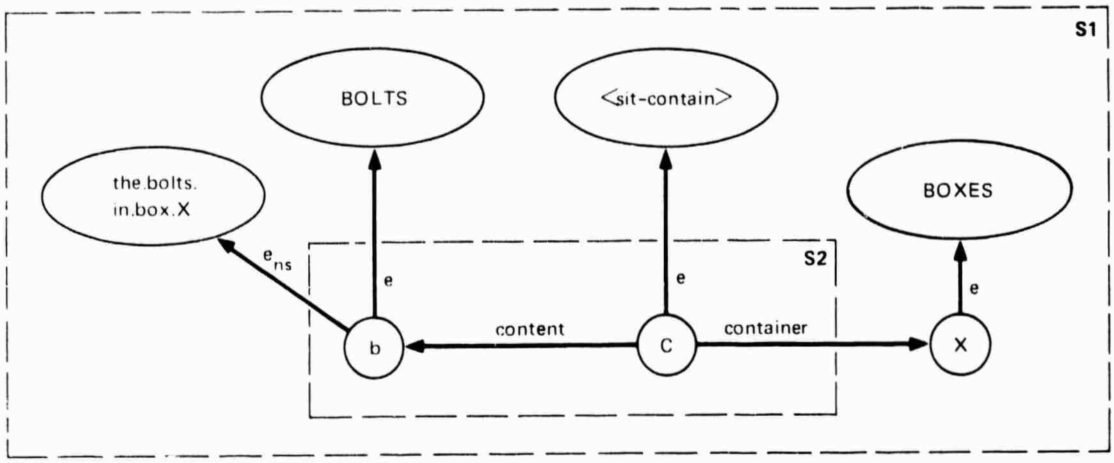


FIGURE V-5 THE NECESSARY AND SUFFICIENT RULE DEFINING "THE BOLTS IN BOX X"

The interpretation of a general statement is that for each assignment of the variables #@Vv to values in their corresponding ranges, there exist entities matching the structure of the #@form. For g this means that for every #b, an element of the.bolts.in.the.box, there exist

#h-C <has.length>

#s-C <sit-screwed:simplistic>

#n-C NUTS

and the relations

#b is the #@object of #h

3/4INCH is the #@measure of #h

#b is the #@mt of #s (i.e., #b is the male-threaded part of #s)

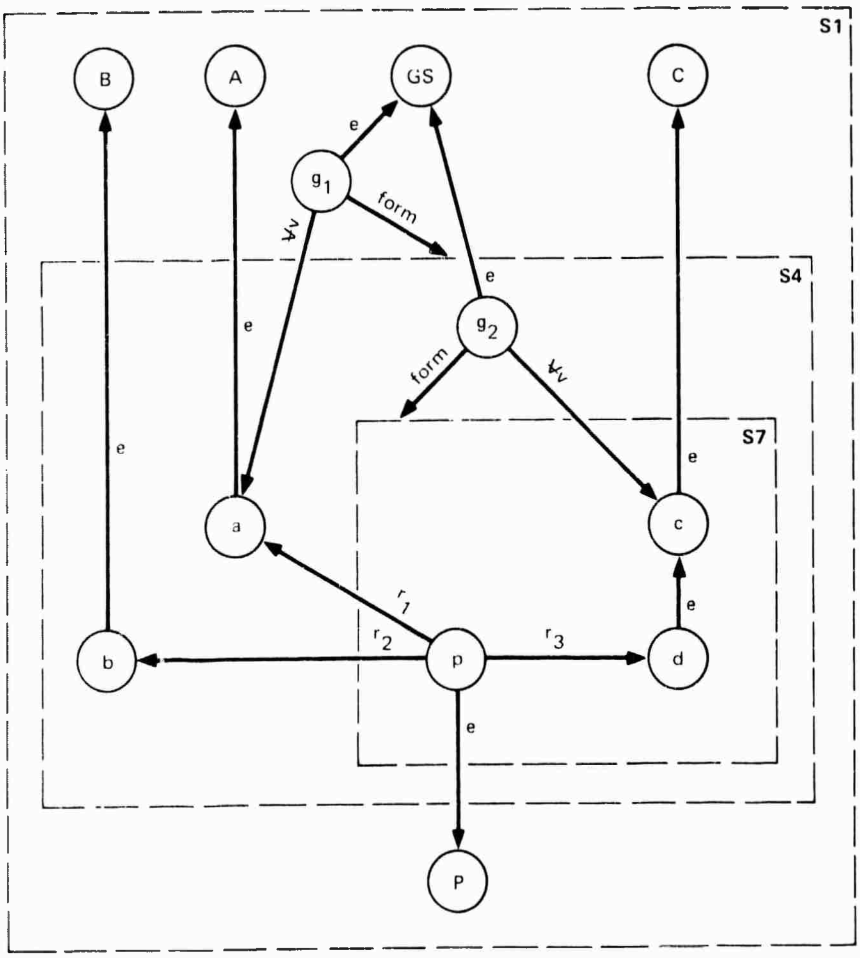
#n is the #@ft of #s.

Thus, the interpretation of g is that for every #b in the.bolts.in.the.box, there exists a situation #h in which the length of OBJECT #b is the MEASURE 3/4 inch. Since '3/4INCH' lies outside space S4, there is only one measure for all the bolts in the box. Furthermore, for each bolt #b there is a nut #n (depending on the individual #b) which is in a situation of being screwed onto #b. (A screwed:simplistic connection may exist only between two threaded objects, one with male threads (the #@mt) the other with female threads (the #@ft). A screwed:simplistic connection may be contrasted with screwed:standard connections in which multiple unthreaded parts are held together by a bolt (or other threaded object) which passes through the unthreaded objects to engage a nut.)

Complex quantifications involving nested scopes may also be encoded by net spaces, as shown abstractly in Figure V-6.

4. Rules and Categories

A convenient method for organizing information in a semantic system is to divide the various objects in the semantic domain (including physical objects, situation objects, and event objects) into a number of categories. Using categories, objects that are somewhat alike become grouped together, allowing similar objects to be thought about and talked about collectively. The scheme is hierarchical in that some categories may be subcategories of more general classes. The lower a class is in the category hierarchy, the more alike its members must be. The likeness arises in that members of each category possess certain common, characterizing properties (such as an association with common attributes or with deep conceptual cases).



SA-3805-30

FIGURE V-6 A COMPLEX ABSTRACT QUANTIFICATION
$(\forall a \epsilon A)(\exists b \epsilon B)(\forall c \epsilon C)(\exists d \epsilon c)[P(a,b,d)]$

The categorical system serves the important purposes of spotlighting similarities among objects and compressing redundant information by recording common information at the category level rather than with the individual. If an object Z is known to belong to some category K, then Z is known to possess the common properties of K's members and the common properties of the members of each of K's supercategories. This ability to encode information at the category level rather than with each individual is of practical importance, because it saves computer memory and because all the elements of a category (perhaps being infinite in number) may not be explicitly known.

For natural language processing, the category system has the important feature that members of the more significant categories (the categories commonly held in the minds of humans) are expressed by the same set of linguistic patterns. As an elementary example, screwdrivers, wrenches, hammers, and saws belong to a category of objects that may be expressed by noun phrases headed by the noun "tool". Various attaching events may be expressed by complete sentences, using the words "attach", "mount", or "fasten" as their central verbs.

Intrinsic in the notion of a category is the notion of a rule that specifies a necessary and sufficient test for category membership. Necessary rules, which all category members must obey, and sufficient rules, which can prove that an object belongs to a given category, are also of importance.

As a simple example of a category and its defining rule, consider the category of bolts in box X. This category is represented by node 'the.bolts.in.box.X' of Figure V-5 with the associated rule being encoded by net space S2. The ens arc lying in space S2 from node 'b' to 'the.bolts.in.box.X' indicates that 'b' represents what may be thought of as an archetypal element of the category. (The label "ens" means "archetypal element, necessary and sufficient.") Any object with the characteristics of b belong to the category and all members of the category have the characteristics of b. As encoded in space S2, the characteristics of b include membership in BOLTS (the set of all bolts) and involvement as the #@content in a containing situation in which box X is the #@container.

In natural language processing, particularly during the parsing phase when surface structures are being translated into nets and when the semantic well formedness of sentences and sentence fragments is being tested, it is important to know what attributes (deep cases) are associated with certain categories of objects (especially with event, situation, and other verb-like categories) and what range of values each attribute may assume. This information is of utility because attributes indicate the types of participants that are involved in particular categories of situations and because there often is a direct mapping from syntactic cases (including prepositional phrases) to these attributes. Knowing the correspondences between surface cases and attributes and knowing the ranges of values for each attribute allow some parses to be rejected on macrosemantic grounds; they also provide a facility for predicting the citing of certain situation participants in the surface language. (This prediction ability is especially important for speech understanding.)

The attribute-range information for a category, collectively referred to as the category's "delineation", may be associated with the category through a delineation rule. A delineation rule is a necessary rule which includes range information about every attribute of the delineated category.

As an example of a delineation rule, consider the delineation of category <to-bolt>, the category of events in which two machine parts are attached by using bolts as fasteners. Delineation information for this category is encoded by the network of Figure V-7. In this network, node '<to-bolt>' is linked to a node 'b' by an ed arc which indicates that b is the delineating "element" of <to-bolt>. Encoded within space S4 is attribute-range information concerning each of the six attributes possessed by members of <to-bolt>. In particular, the rule encoded by space S4 indicates that a bolting event involves an #@actor taken from the set of INTELLIGENT.ANIMATE.OBJECTS, a #@minor-p and a #@major-p taken from the set of MACHINE.PARTS, a set of #@fasteners taken from the set of BOLT/NUTS (a bolt/nut is a bolt and a nut that work together to form a single fastener), a #@tool taken from the set of TOOLS (which includes hands and fingers), and a #@time taken from the set of TIME.INTERVALS.

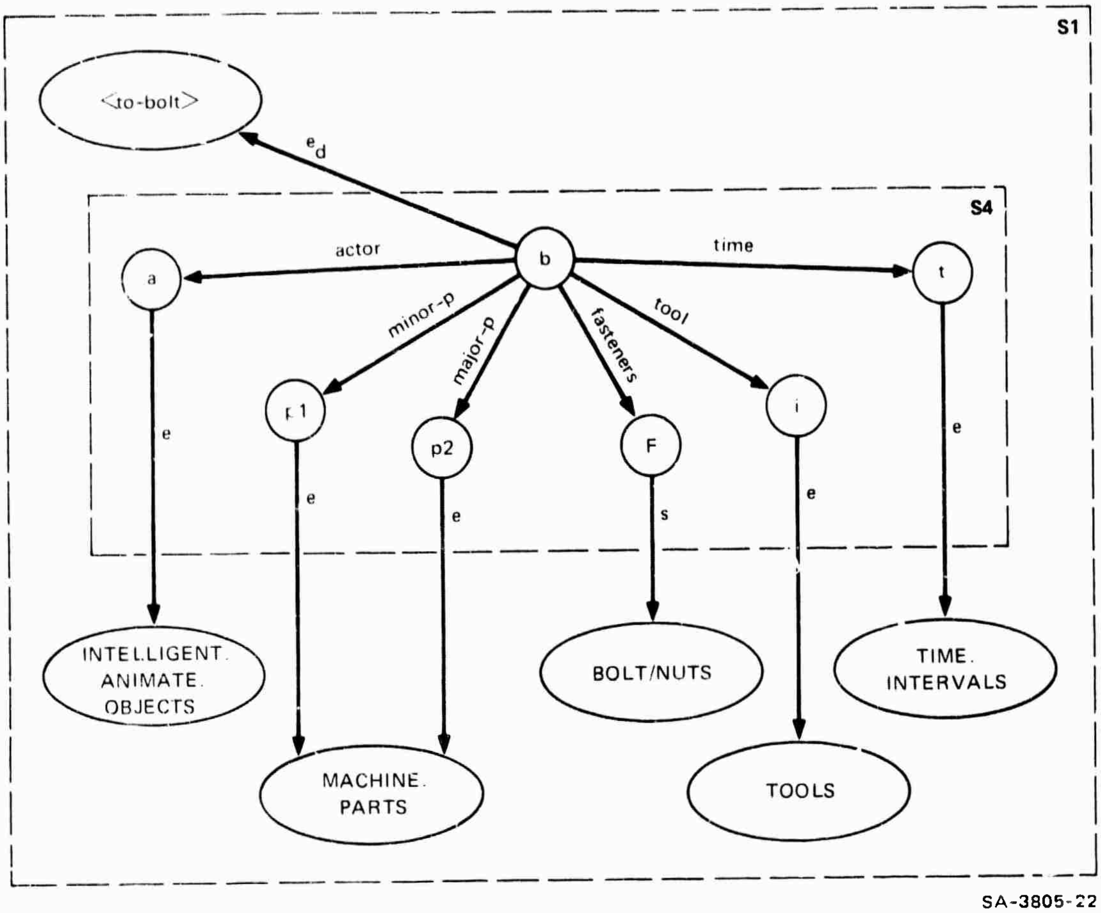


FIGURE V-7 DELINEATION OF <TO-BOLT>

Given the two sentences

I bolted the pump to the base plate WITH 1 INCH BOLTS.

I bolted the pump to the base plate WITH THE WRENCH.

the delineation of <to-bolt> may be used to determine that the WITH phrase in the first sentence supplies the #@fasteners case while in the second sentence it supplies the #@tool case.

The delineation rule of Figure V-7 shows all delineation information concerning <to-bolt> to be encoded in a single rule linked directly to the category. In practice, categorical information is almost always distributed among many points in the categorical hierarchy. To see how information is distributed at various levels, consider the hierarchy of <to-attach> events which is exhibited in Figure V-8. The most general category in the hierarchy is category U, the universal set. Even U has a delineation since all objects (including events and situations) exist over some (possibly one-point or infinite) time interval. A subset of U is <to-attach>, the set of all attaching events of any nature whatever. Members of <to-attach> inherit the time attribute from supercategory U and add two additional attributes, #@parts and #@actor, of their own. In general, each attaching event involves a set of #@parts that an #@actor binds together in some way.

Two subcategories of <to-attach> are shown in the figure. The first is <to-screw:simplistic>, which is the set of events in which two threaded objects, one (#@mt) with male threads, the other (#@ft) with female threads are engaged by twisting. Notice that the delineation rule of this category shows that the #@mt and the #@ft are both elements of the #@parts. The cardinality of #@parts is at most two (but could be one as for a garden hose with one end attached to the other).

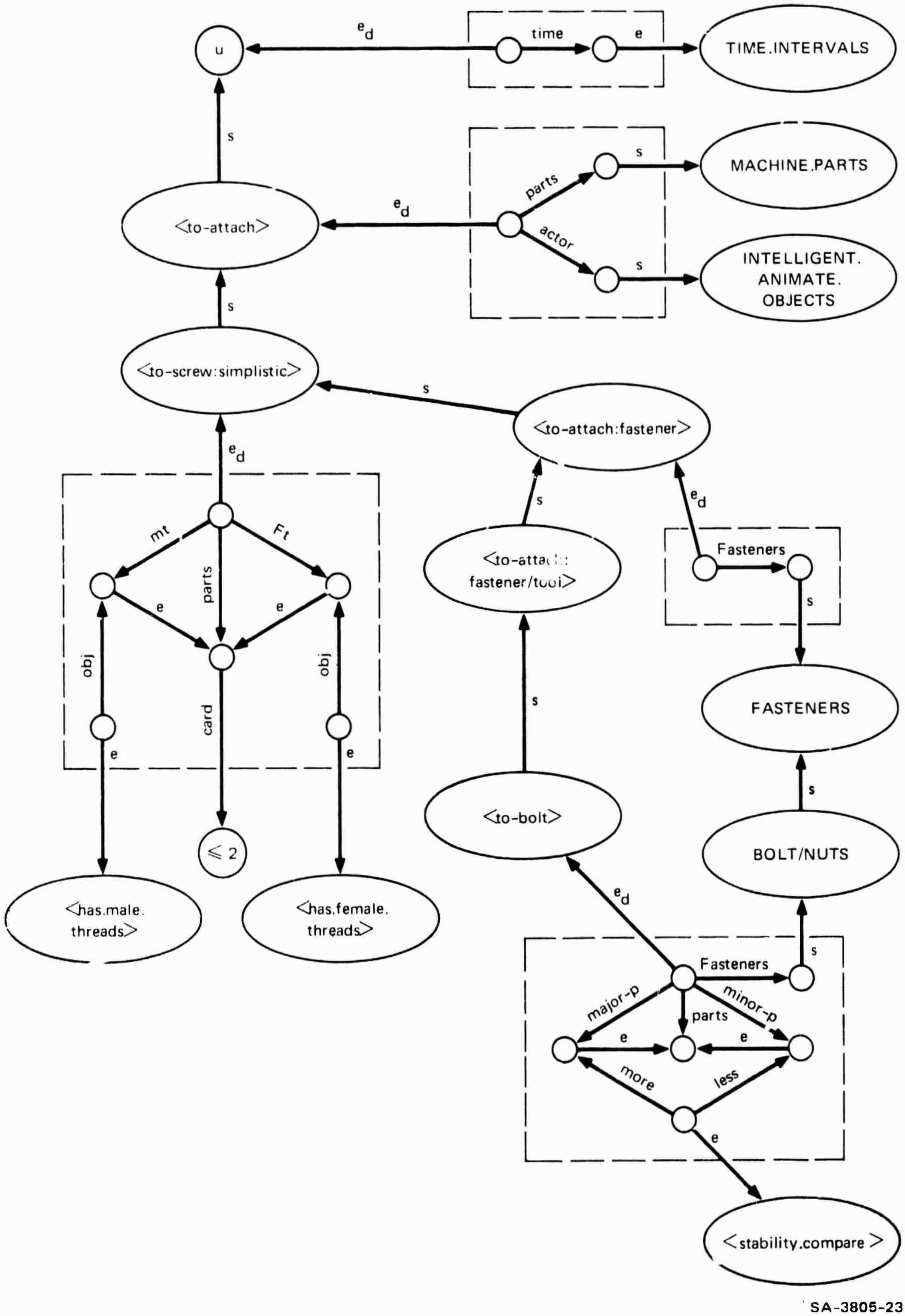


FIGURE V-8 THE <TO-ATTACH> FAMILY

A second subcategory of <to-attach> is <to-attach:fastener>, the category of fastening events in which the #@parts are attached with fasteners. (Screwing a lightbulb into a socket requires no fasteners and is a simplistic screwing event. Nailing a sign to a post requires a nail as a fastener.) The delineation of <to-attach:fastener> simply adds the attribute of @fasteners.

Category <to-bolt> is a subcategory of <to-attach:tool> which is a subcategory of <to-attach:fastener>. The delineation of <to-bolt> shown in Figure V-8 indicates how the #@major-p and the #@minor-p are related to #@parts and to each other. Furthermore, the #@fasteners used by bolting events are restricted to be bolt/nuts as opposed to any type of fastener. Linkage to a process automaton which indicates the sequence of changes characterizing a bolting event might also be included with the category information but has been omitted here for simplicity.

5. Abstraction

Since a user may think at varying levels of detail, it is important in our computer consultant task domain for the semantic system to be able to encode information at multiple levels of abstraction and have some capability for jumping from one level to another. Figure V-9 shows one way in which net partitioning may be used to encode an attaching event A at two levels of detail. By viewing the network from the vantage of

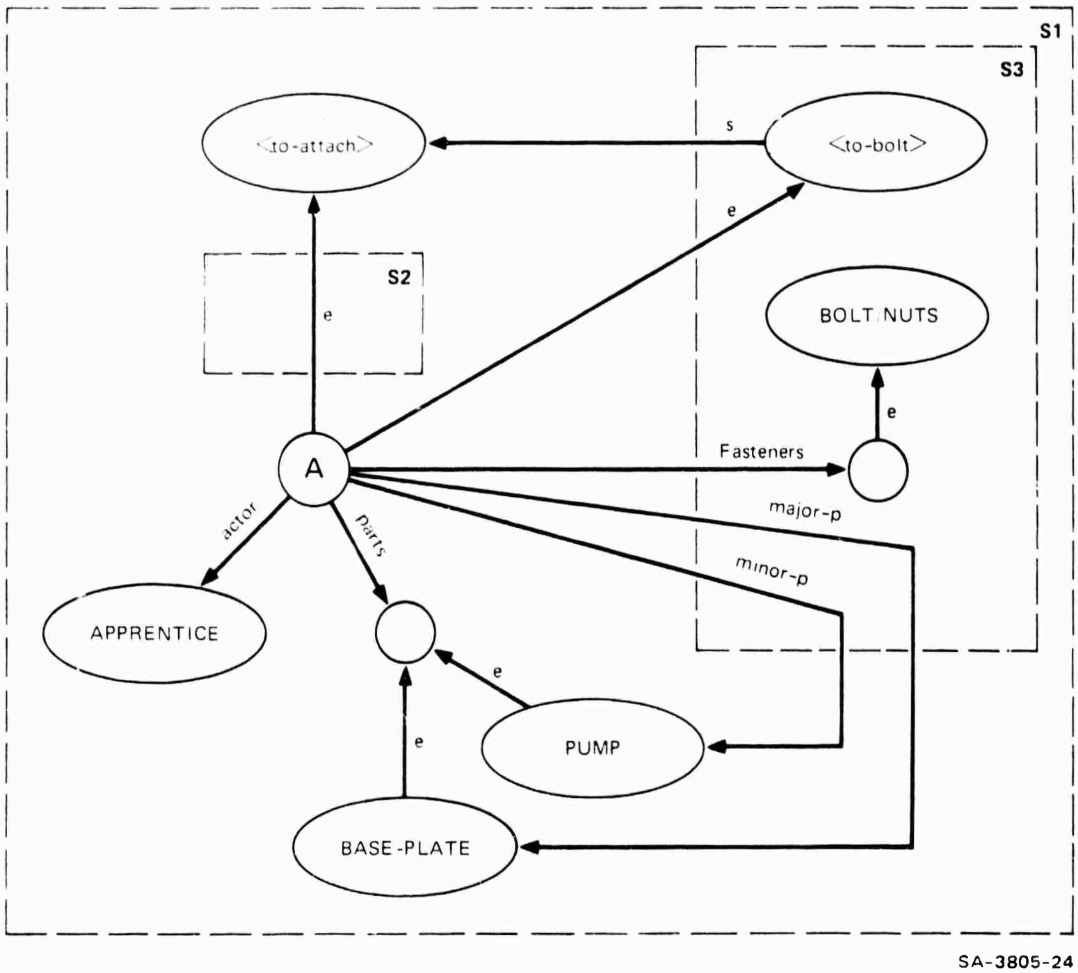


FIGURE V-9 VIEWING A BOLTING AT TWO LEVELS OF DETAIL

space S2 (which lies below S1 in the ordering and is a sister space to S3), A is seen to be an element of <to-attach> since the e arc lying in S2 is visible. Since the information lying in S3 is invisible from S2, A appears to have only an #@actor and a set of #@parts and is not seen to entail #@fasteners. From S3 the same event may be viewed with more detail. First, the e arc from A to <to-attach> is invisible, and A is thus seen as an element of

<to-bolt>, a subset of <to-attach>. Furthermore, at this finer level of detail, the *@fasteners used in the attaching (bolting) event are visible (as are tools and other elements, which are omitted from the figure for simplicity).

6. Processes

A very important aspect of the computer consultant task domain is that of change. Since sequences of change tend to follow certain regular patterns, it is convenient to organize the recurring sequences of change into categories, grouping similar sequences together. Each category of sequential change is tantamount to an event category, the members of which may be thought of as individual enactments of a common plot or script that encodes a generalized pattern of change. For example, every event of tightening bolts follows the plot consisting of finding a wrench, putting the wrench on the bolt, twisting the bolt clockwise, and so on. Each enactment casts different participants in the various roles but follows the same basic pattern.

Since the members of a particular event category may be distinguished as exactly the instantiations of sequential change that follow some particular script, the script itself forms the basis for a rule defining the event category.

During the past year we have been considering ways to encode process scripts in semantic networks for use in language processing. The procedural nets developed by the planning group of the SRI Computer Based Consultant Project (Sacerdoti, forthcoming; Nilsson, 1975) are a representation of process knowledge and we anticipate the eventual merger of procedural and semantic networks. However, since procedural nets were not designed with language processing in mind, we have considered process automata (see Hendrix, forthcoming) as a possible alternative. A process automaton is a section of semantic network that resembles a Mealy machine or an augmented finite-state transition network (AFTN) system (Woods, 1970). While the AFTN model was developed as a programming structure to describe the process of parsing language, the process automaton has been developed as a data structure for describing the processes (prototypal plots) cited by language and may be regarded as a parsing grammar that interprets (or generates) a sequence of changing conditions rather than a sequence of words. If a path can be found through a process automaton network for a given sequence of changes, the sequence is accepted as a 'word' (an enactment) in the 'language' (category of events) defined by the 'grammar' (process automaton).

C. The Initial Implementation

Many of the ideas concerning net partitioning that were presented in the previous section were either conceived as a result of or tempered by our experiences with a network based semantic system that was designed, built, and tested during the summer and fall of 1974. The concepts on which this system is based have been modified during testing and evaluation. Nevertheless, this section presents the system in its original form. Changes, which are in process, are discussed in the following section.

In overview, the system is built around four major constructs:

(1) A network data structure encoding the basic task-domain knowledge of the system.

(2) A network 'scratch pad' for use in building network representations of input utterances and their component parts,

(3) An 'intermediate language' between surface English and network notation, intended for use as an aid in discourse analysis.

(4) A battery of semantic composition routines that are called by the parser to test the semantic compatibility of phrase constituents and to build semantic representations (on the scratch pad) of complete phrases given the semantic representations of component parts.

Routines for querying the data base to retrieve answers to user questions and routines for making semantic predictions are planned but are awaiting revisions to the basic network manipulation routines.

The discussion of this section is presented in two parts. The first part considers our methods of encoding domain-specific knowledge for the data management task, while the second part presents a set of translation examples that illustrate the use of the scratch pad, intermediate language, and composition semantic routines.

1. The Knowledge Network

The top level of the network encoding of the data base for our submarine protocol experiments is shown in Figure V-10. This network follows closely the conventions presented in the previous section. The top node in this network is 'UNIOBJS', the node representing the universal set of objects. Major subsets of UNIOBJS include RELATIONS, MEASURES (a measure is a number/unit combination such as 30 knots), LEGAL.PERSONS (a legal person is an entity such as a person, corporation, or government that may enter into contracts) and PHYSOBJS (the set of physical objects).

All the information in the data base concerns submarines and the relationships in which submarines are participants. Similar relationships (e.g., all ownerships) are collected into subsets of RELATIONS. For example, the set OWN.RELS of ownership relationships appears in the data base and is delineated by the net space labeled "own". This delineation indicates that an element of OWNS.RELS has an #@owner taken from the set of LEGAL.PERSONS and an #@ownee taken from the set of PHYSOBJS. (Time arcs are not included since the data base is assumed to be

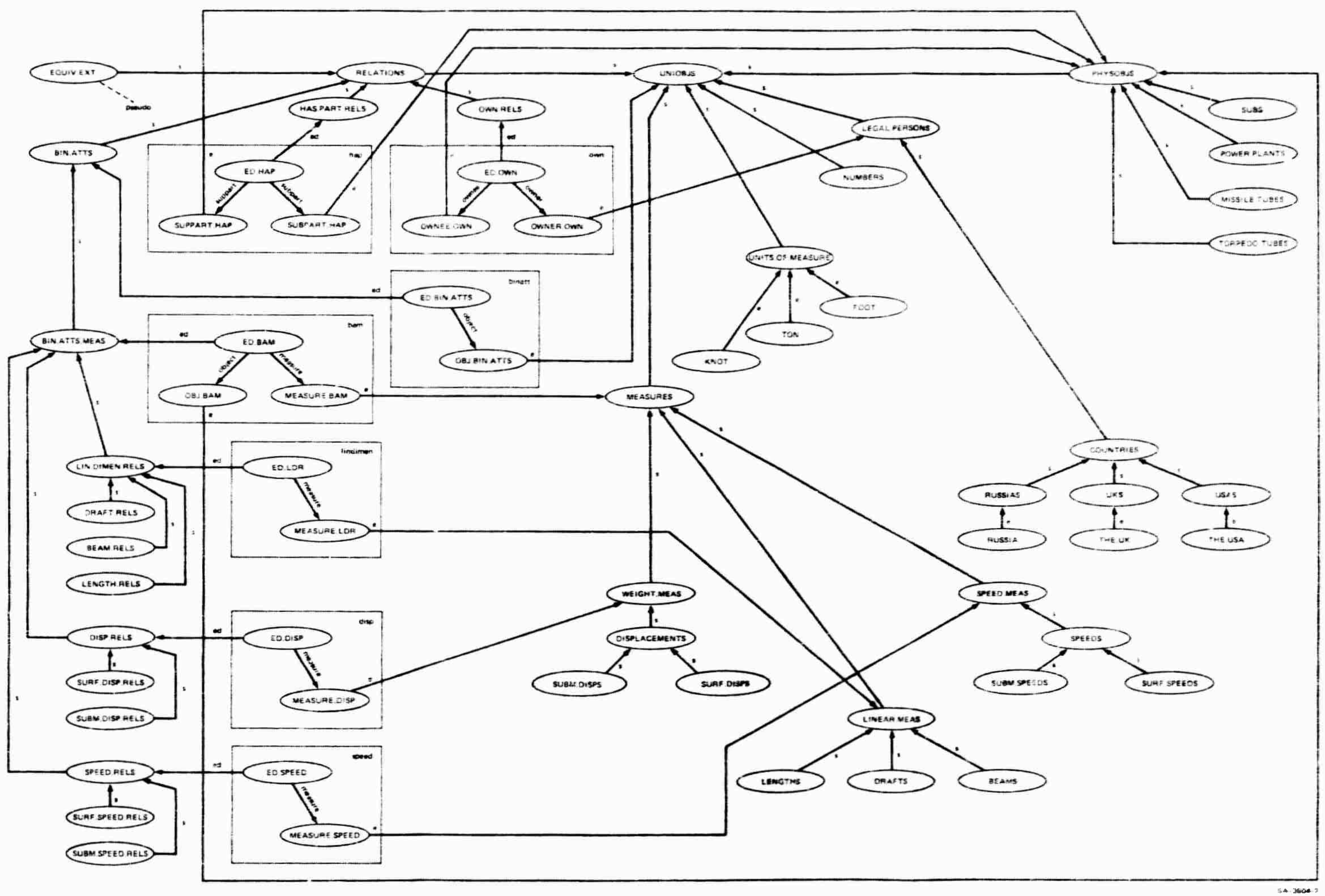


FIGURE V-10 TOP LEVEL KNOWLEDGE SPACE

static and all facts recorded are assumed to be true at the current time. The network encoding of HAS.PART.RELS (has as part relationships) is similar to OWN.RELS. The constituents of a HAS.PART.RELS relationship are a #@suppart ("sup" is taken from "super") and a #@subpart, both taken from set PHYSOBJS. (#@subpart is a part of #@suppart.)

Set BIN.ATTS is the set of so-called binary-attribute relationships. Members of this class are typically expressed by the construct

The X of the Y is Z

and cannot be expressed by a verb form of X. Thus "The speed of the sub is 40 knots," qualifies speed relationships as BIN.ATTS. However, since "The owner of the sub is the U.S." can be stated using the verb form "to-own" of "owner" (as in "The U.S. owns the sub."), ownerships are not considered to be BIN.ATTS. As encoded by the rule of the space labeled "binatt", each member of BIN.ATTS has an #@object taken from the universal set.

BIN.ATTS.MEAS is the subset of BIN.ATTS whose members relate an @object to some #@measure. If the #@measure is taken from LINEAR.MEAS (is a linear measure), then the relationship may be an element of LIN.DIMEN.RELS, the set of relationships of linear dimensions. A subset of LIN.DIMEN.RELS is LENGTH.RELS, the set of relationships whose members relate an object to the measure of its length.

In addition to the top level information shown in Figure V-10, the knowledge base network also includes more specific pieces of information such as those shown in Figure V-11. Information encoded in Figure V-11 may be used to answer specific questions concerning Lafayette class submarines. As may be seen by the network, LAFAYETTES (the set of Lafayette subs) is a subset of SUBS and the VON-STEUBEN is a particular Lafayette.

Associated with LAFAYETTES is the necessary-type rule encoded by the space labeled "lafe". This rule indicates that all Lafayette subs have the properties of EN.LAFE, the necessary archetypal element of LAFAYETTES. In particular, every Lafayette is owned by the U.S. (node 'P'), has a surface-displacement of 8200 tons (node 'SD'), and has four torpedo tubes as subparts (node 'H'). The nodes 'THE.US' and 'M' lie outside space lafe, since all subs have the same owner and surface displacement. But node 'T' lies inside the space since each sub has its own set of torpedo tubes. The arc labeled "subpart*" from 'H' to 'T' is a kind of shorthand meaning that every element of set T is a #@subpart of EN.LAFE. The set itself is not a subpart, but each of its members is. This shorthand is now being replaced by quantified statements of the type described in the previous section.

2. Translation Examples

Rather than discuss the network scratch pad, the intermediate language, and the composition routines separately,

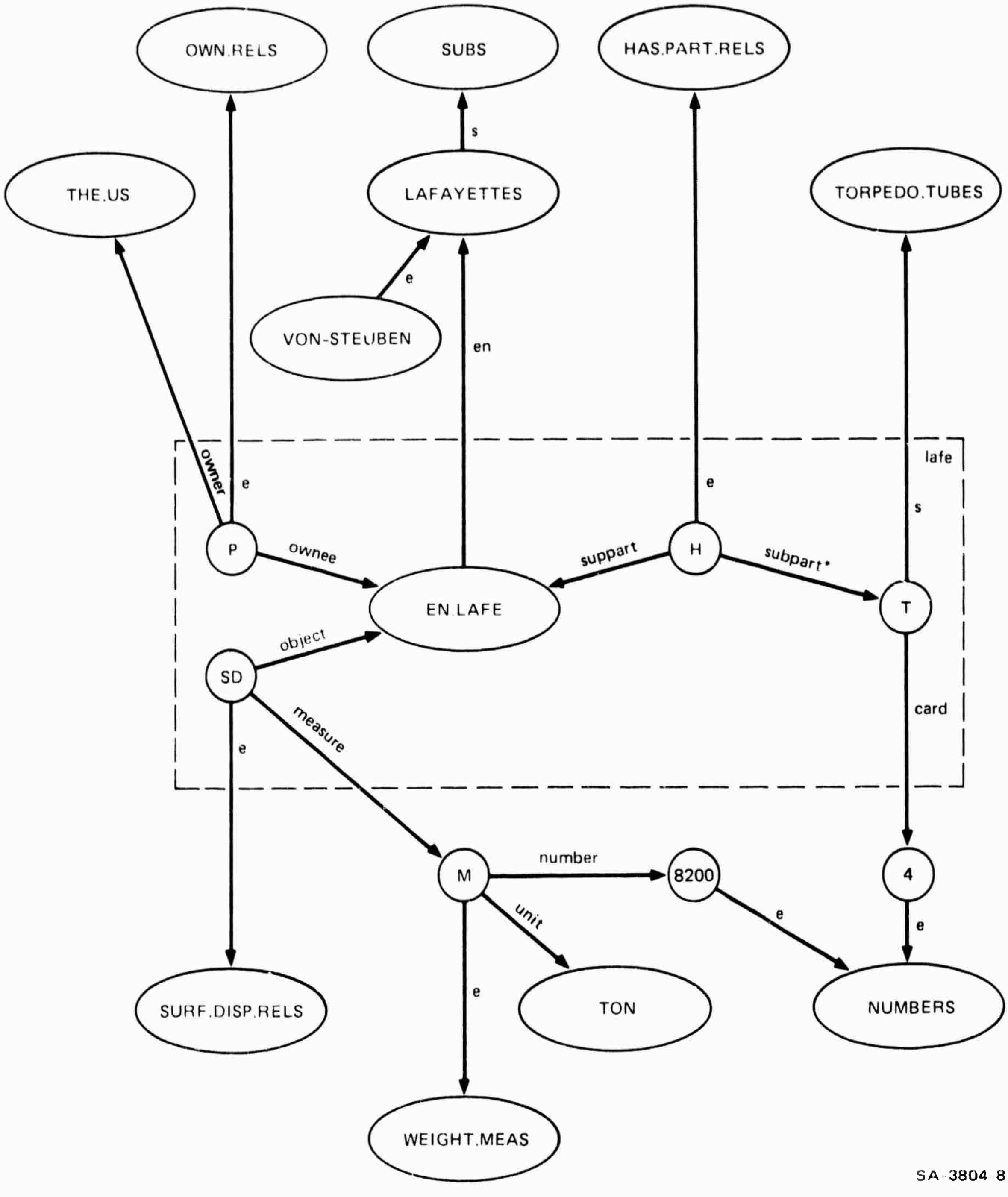


FIGURE V-11 PARTIAL DATA FOR LAFAYETTE SUBMARINES this section describes the three concurrently through examples of how inputs are translated into their network representations. In building up a semantic interpretation of a phrase, the composition routines derive information from the semantic attributes of the constituents of the phrase. Ultimately, the most primitive attributes, those associated with individual words or phonemes, are recorded in the lexicon. The word-semantics for each of the dozen words in the examples that follow are presented in Figure V-12.[3] The meaning of these partial lexical entries will be presented through the discussion of the examples.

a. Example 1

As the first example, consider the interpretation of the utterance

The U.S. owns one of the four subs.

Although this is a contrived sentence that has not appeared in our protocol experiments, it will serve to point out the basic features of our translation and encoding systems while postponing side issues.

------
[3] A listing of the lexicon currently in use is presented in Appendix A. The entries of Figure V-12 reflect lexical entries as they appeared in the initial implementation.

Figure V-12 Semantic Information from Selected Lexical Entries

```
does - DO
[(TYPE DO)(NBR S)]

four - DIGIT
[(TYPE DIGIT)(DIGTYP (1 2 3))(NUM 4)]

is - BE
[(TYPE BE)(NBR (SET M S))]

Lafayette - N
[(TYPE N)(SUPSET 'LAFAYETTES')(CMU COUNT)(NBR S)]

of - PREP
[(TYPE PREP)]

of - TOKEN
[(TYPE TOKEN)]

one - DIGIT
[(TYPE DIGIT)(DIGTYP 1)(NUM 1)]

own - V
[(TYPE V)(SUPSET 'OWN.RELS')(PDGM PG.OWN)
 (MANDATORY (OWNER OWNEE))]

sub - N
[(TYPE N)(SUPSET 'SUBS')(CMU COUNT)(NBR S)]

surface-displacement - N
[(TYPE N)(SUPSET 'SURF.DISPS')(CMU COUNT)(NBR S)(NBR S)
 (INVERSIONS [[(TYPE VP)(SUPSET 'SURF.DISP.RELS')
               (PDGM BIN.ATT)
               (CASES [(MEASURE *)])]])]

the - ART
[(TYPE ART)(DET DEF)(NBR (SET M PL S))]

US - N
[(TYPE N)(SUPSET 'USAS')(CMU COUNT)(NBR S)]

what
[(E?)(TYPE DET)(SUBTYPE (SET 1 2))(DET ?)
 (NBR (SET M PL S))]
```

The parse tree of this example utterance is shown in Figure V-13, where the symbols enclosed in ellipses are the designations of the composition rule definitions used to parse the utterance. (See Appendix A and the discussion in Section IV, The Language ule contains a semantic part

which builds a semantic representation of the resultant phrase from t semantic representations of its components. The semantic representation of an utterance component is either an expression in the intermediate language (which consists of a list of attribute-value pairs) or an expression in the intermediate language accompanied by a network structure. The intermediate language representations (ILRs) of the various phrases composing the example utterance are listed in Figure V-12 (for primitive lexical entries) and in Figure V-14 (for components produced through the application of rules). Entries in both figures are alphabetized. The network representations of the relevant components are presented in the various subfigures of Figure V-15.

Since the parser is capable of initiating the parsing of subphrases anywhere within an utterance, the order in which the subtrees of the total parse tree are encountered is irrelevant. Thus, the discussion of how the composition semantics operates can begin with the word "subs" at the far right of the utterance. (With the inclusion of a word spotter in the system's acoustic component, it would be reasonable for processing to start with that word, since it contains two sibilants, which re

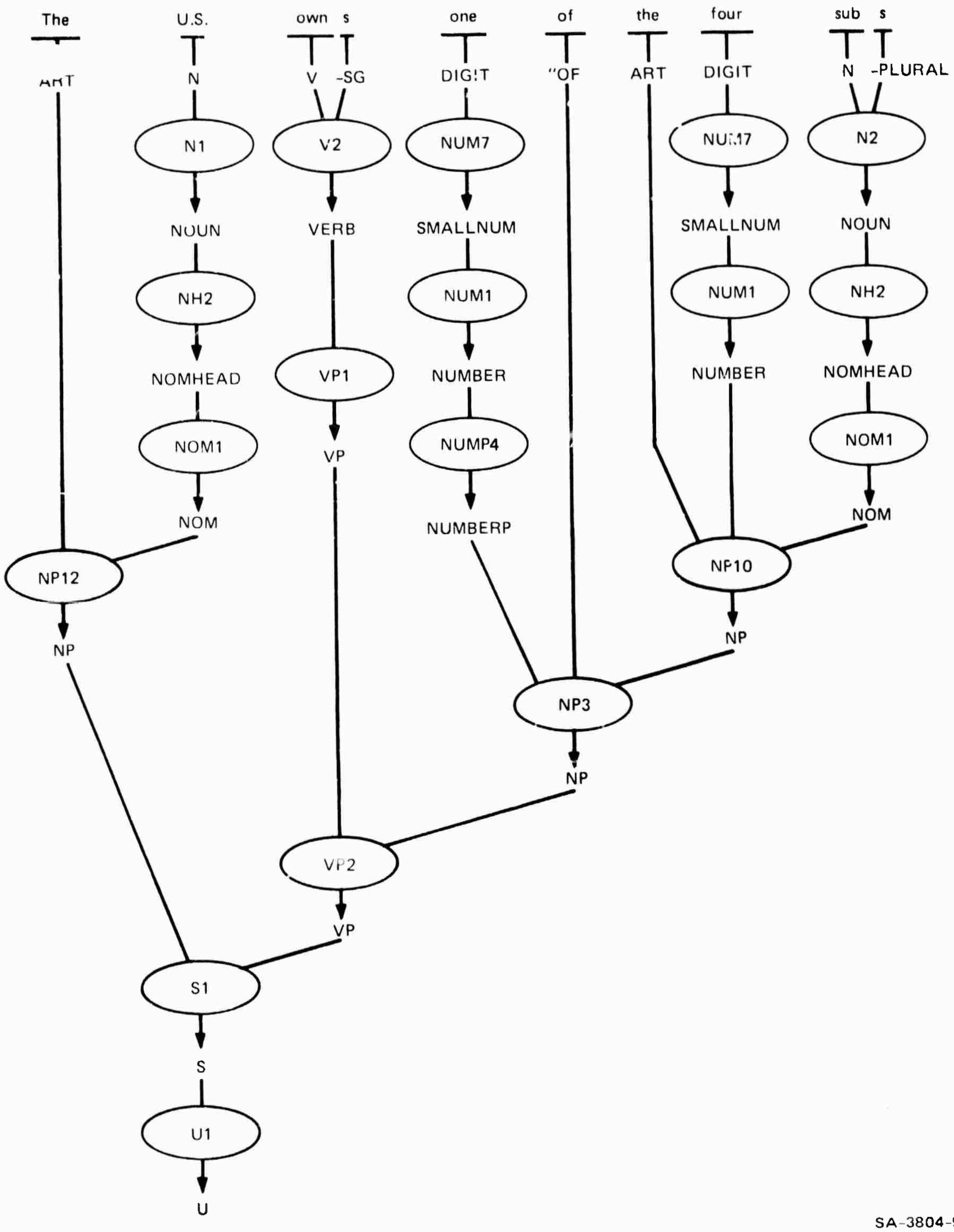


FIGURE V-13 PARSE TREE OF "THE U.S. OWNS ONE OF THE FOUR SUBS"

Figure V-14 Intermediate Language Semantics of Phrases from "The U.S. owns one of the four subs"

```
four - SMALLNUM
[(TYPE SMALLNUM)(NUM 4)(NBR PL)]

four - NUMBER
[(TYPE NUMBER)(NUM 4)(NBR PL)(NET '4')]

one - SMALLNUM
[(TYPE SMALLNUM)(NUM 1)(NBR S)]

one - NUMBER
[(TYPE NUMBER)(NUM 1)(NBR S)(NET '1')]

one - NUMBERP
[(TYPE NUMBERP)(NUM 1)(NBR S)(NET '1')]

one of the four subs - NP
[(TYPE NP)(NUM 1)(NBR S)(NET 'G2')
 (SUPSET* [(TYPE NP)(SUPSET 'SUBS')(CMU COUNT)
          (NBR PL)(NET 'G1')(NUM 4)(DET DEF)])]

owns - VERB
[(TYPE VERB)(SUPSET 'OWN.RELS')(PDGM PG.OWN)
 (MANDATORY (OWNER OWNEE))(NBR (SET M S))]

owns - VP
[(TYPE VP)(SUPSET 'OWN.RELS')(PDGM PG.OWN)
 (MANDATORY (OWNER OWNEE))(NBR (SET M S))(NET 'G4')]

owns one of the four subs - VP
[(TYPE VP)(SUPSET 'OWN.REL')(PDGM PG.OWN)
 (MANDATORY (OWNER OWNEE))(NBR S)(NET 'G4')
 (PDGM.MESSAGE NIL)
 (CASES [(OWNEE [(TYPE NP)(NUM 1)(NBR S)(NET 'G2')
                (SUPSET* [(TYPE NP)(SUPSET 'SUBS')
                          (CMU COUNT)(NBR PL)
                          (NET 'G1')(NUM 4)
                          (DET DEF)])])])]

subs - NOUN
[(TYPE NOUN)(SUPSET 'SUBS')(CMU COUNT)(NBR PL)]
```

Figure V-14 Intermediate Language Semantics of Phrases from "The U.S. owns one of the four subs" (concluded)

```
subs - NOMHEAD
[(TYPE NOMHEAD)(SUPSET 'SUBS')(CMU COUNT)(NBR PL)
 (NET 'G1')]

subs - NOM
[(TYPE NOM)(SUPSET 'SUBS')(CMU COUNT)(NBR PL)(NET 'G1')]

the four subs - NP
[(TYPE NP)(SUPSET 'SUBS')(CMU COUNT)(NBR PL)(NET 'G1')
 (NUM 4)(DET DEF)]

The US - NP
[(TYPE NP)(SUPSET 'USAS')(CMU COUNT)(NBR S)(NUM 1)
 (NET 'G3')(DET DEF)]

The US owns one of the four subs - S and U
[(TYPE S)(SUPSET 'OWN.RELS')(PDGM PG.OWN)
 (MANDATORY (OWNER OWNEE))(NBR S)(NET 'G4')
 (PDGM.MESSAGE NIL)
 (CASES [(OWNER [(TYPE NP)(SUPSET 'USAS')(CMU COUNT)
                 (NBR S)(NUM 1)(NET 'G3')
                 (DET DEF)])
         (OWNEE [(TYPE NP)(NUM 1)(NBR S)(NET 'G2')
                 (SUPSET* [(TYPE NP)(SUPSET 'SUBS')
                           (CMU COUNT)(NBR PL)
                           (NET 'G1')(NUM 4)
                           (DET DEF)])])])]

US - N
[(TYPE NOUN)(SUPSET 'USAS')(CMU COUNT)(NBR S)]

US - NOMHEAD
[(TYPE NOMHEAD)(SUPSET 'USAS')(CMU COUNT)(NBR S)(NUM 1)
 (NET 'G3')]

US - NOM
[(TYPE NOM)(SUPSET 'USAS')(CMU COUNT)(NBR S)(NUM 1)
 (NET 'G3')]
```

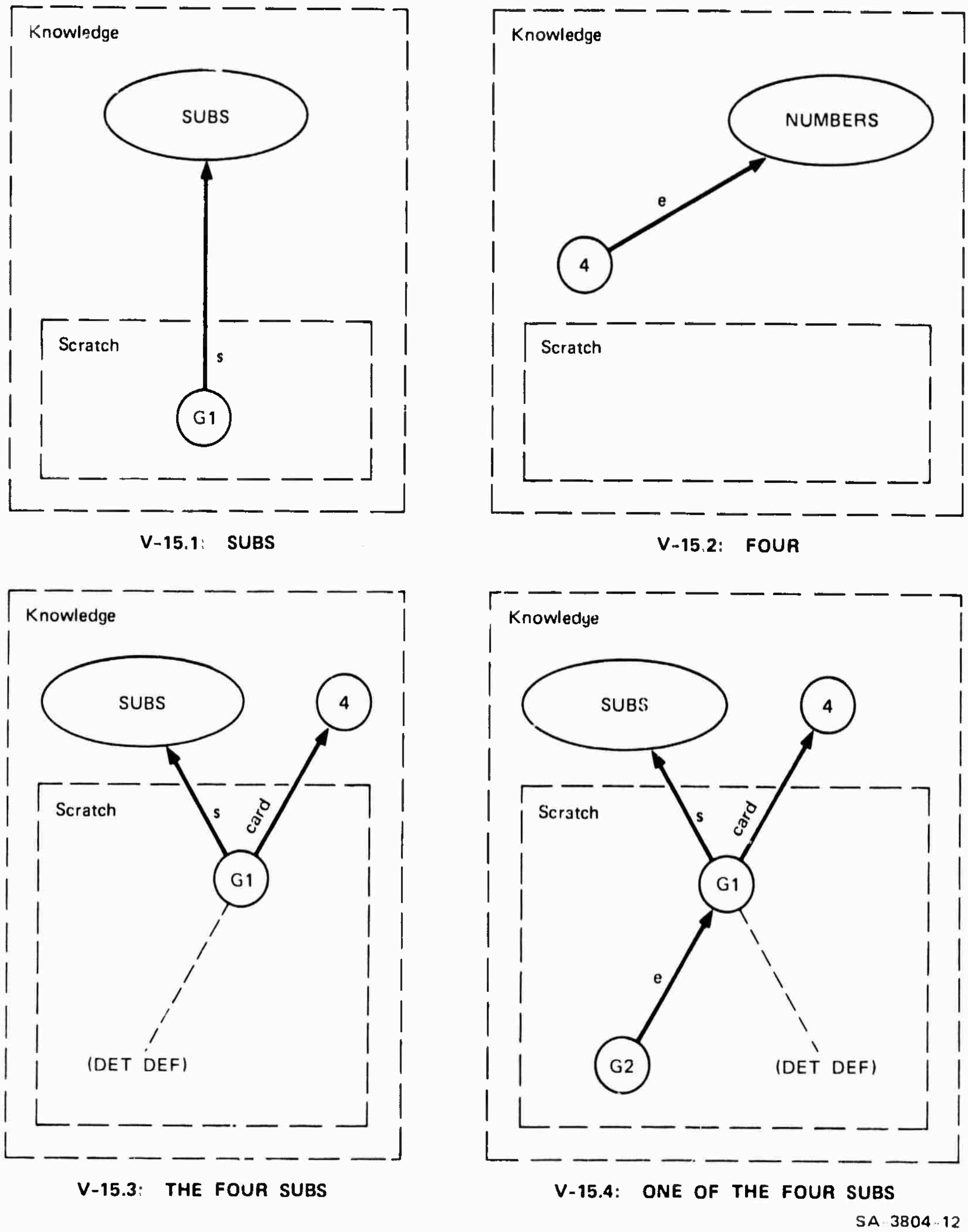
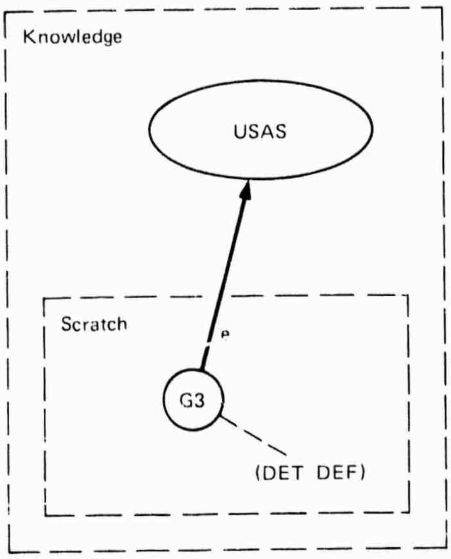
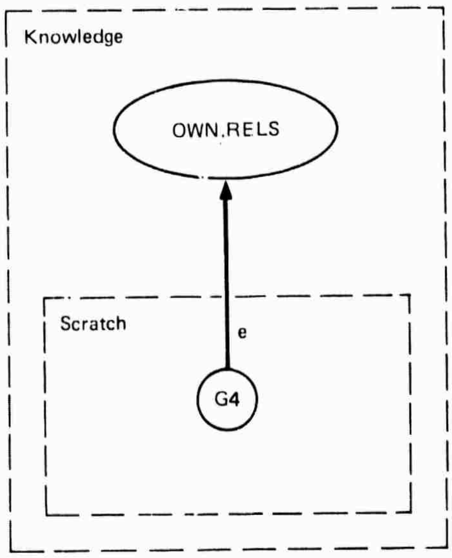


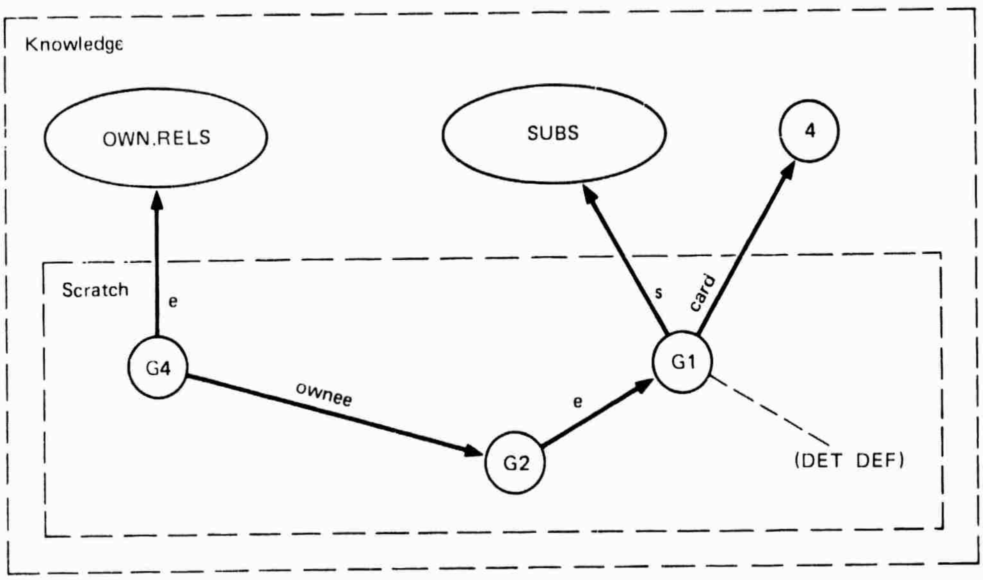
FIGURE V-15 NET SEMANTICS OF PHRASES IN "THE U.S. OWNS ONE OF THE FOUR SUBS"



V-15.5: THE U.S.



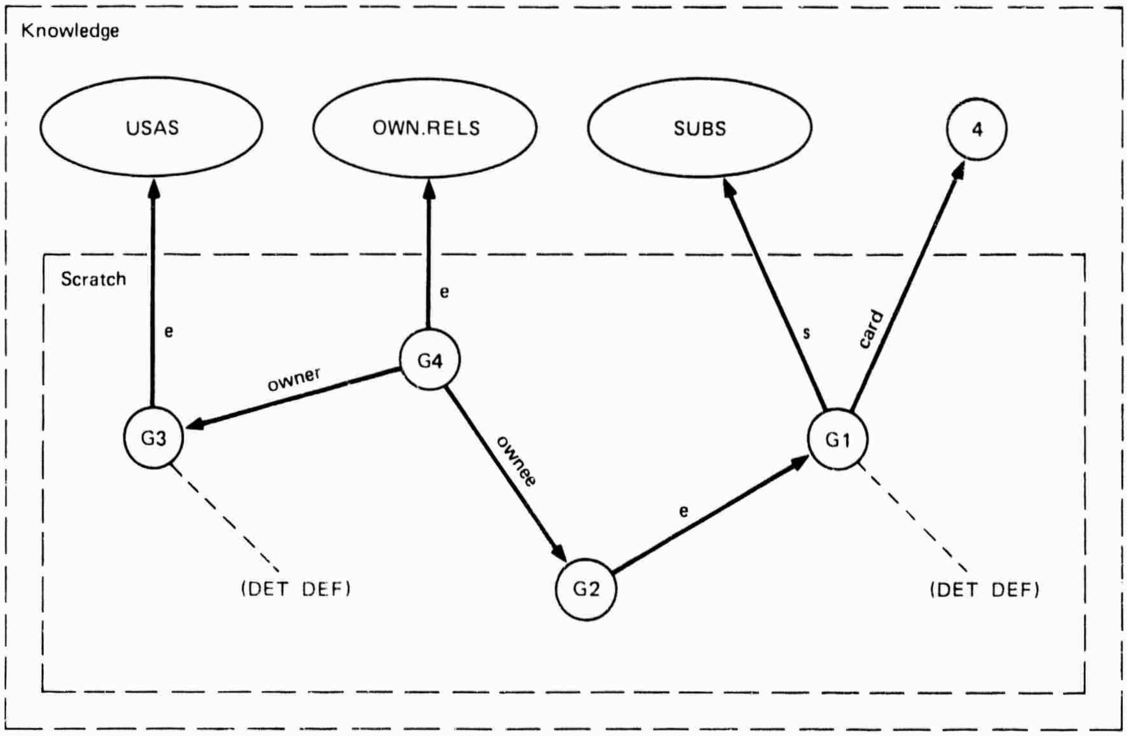
V-15.6: OWNS



V-15.7: OWNS ONE OF THE FOUR SUBS

SA-3804-13

FIGURE V-15 NET SEMANTICS OF PHRASES IN "THE U.S. OWNS ONE OF THE FOUR SUBS" (Continued)



V-15.8: THE U.S. OWNS ONE OF THE FOUR SUBS

SA-3804-14

FIGURE V-15 NET SEMANTICS OF PHRASES IN "THE U.S. OWNS ONE OF THE FOUR SUBS" (Concluded) relatively easy to locate, and since it has a high relative frequency in discourses within this task domain.)

From the lexical entry of Figure V-12, the stem "sub" is seen to have the semantics

[(TYPE N) (SUPSET 'SUBS') (CMU COUNT) (NBR S)]

From this entry, the TYPE of "sub" is seen to be N, indicating that "sub" is a noun stem. That is, "sub" may be combined with a suffix (including the empty suffix) to produce a NOUN. The CMU (i.e., Count-Mass-Unit) of "sub" is COUNT, indicating that subs are countable objects. The NBR (NumBeR) of "sub" is S, indicating that, if only the empty suffix is added to "sub", the result will be a singular noun. The SUPSET (superset) of a linguistic entity (noun, verb, adjective) is a node in the semantic network that represents the set containing all the objects named by the entity. The stem "sub" names members of the set SUBS, the set of all submarines, represented by node 'SUBS' of the semantic net. (As other examples, the SUPSET of "own" is 'OWN.RELS', the set of all relationships of ownership. The SUPSET of "buy" would be 'EXCH', the set of all canonical exchange events.)

Rule N2 is used to combine an N such as "sub" with a pluralizing suffix (such as "s"). The result of the application of rule N2 (as seen in Figure V-14) is a constituent of TYPE NOUN with NBR changed from S to PL (for plural).

Working down the parse tree, the NOUN "subs" is next transformed into the NOMHEAD "subs" by the application of rule NH2. While this transformation adds only one new attribute-value pair to the intermediate language representation of "subs", the step is crucial to the translation process (and to the reader's understanding), because it is in this step that a representation of "subs" first appears in the net. This entry is made on the network scratch pad, which is actually a net space lying just below a space that encodes general system knowledge, which is called the 'knowledge space'. The information recorded in this subordinate scratch space is invisible from the knowledge space and thus cannot become confused with the general knowledge in the system.

The entry is made in the following way. First, a new node is created in the scratch space and given an arbitrary name, such as G1. In accordance with the principle that utterances are understood in relation to existing knowledge, this new node must be linked to concrete information in the knowledge space. The attributes SUPSET, CMU, and NBR of the intermediate language are used to determine what this link should be. For "subs", the linkage, as shown in Figure V-15.1, is an s arc from 'G1' to 'SUBS'. Node 'SUBS' of the knowledge space is used because it is the value of attribute SUPSET. The s (or subset) link is used for "subs", because the CMU attribute has value COUNT and NBR has value PL, meaning that "subs" represents a set of countable objects that is a subset of SUBS. Had the NBR been S (for singular), an e (element of) arc would have been used. For a CMU of MASS, a mass.subset arc would have been used.

The new attribute-value pair introduced into the ILR by rule NH2 is the pair (NET 'G1'), indicating that the NOMHEAD "subs" is represented in the network by node 'G1'.

The next transformation on "subs" is accomplished by rule NOM1, which (for this example) does nothing but change the TYPE to NOM. Before this NOM may be converted into an NP through rule NP10, the DIGIT "four" must be transformed into a NUMBER.

The lexical entry for "four", Figure V-12, includes the attribute-value pair (DIGTYP (1 2 3)). The DIGTYP is used in determining how a DIGIT may be combined to form larger numbers. Type 1 DIGITS may stand alone as numbers. Hence "four" is a number all by itself while "twen", with (DIGTYP 3), and "thir", with (DIGTYP (2 3)), may not be. Type 2 DIGITS may form teens; hence "fourteen" and "thirteen" but not "twenteen". Type 3 DIGITS may form a DIGTY such as "forty" and "thirty". The DIGIT "one" is type 1 only and hence may not form "oneteen" or "onety".

Since DIGIT "four" is of type 1, it may be converted into a SMALLNUM by rule NUM7 and then into a NUMBER by rule NUM1. Nodes corresponding to numbers are not initially in the net but are generated as needed. All rules that produce a NUMBER check to see if the NUMBER so produced has been encoded in the knowledge space of the semantic network. For the number "four", a check is made to see if a node '4' exists that has an e arc to node 'NUMBERS'. If such a node and arc do not exist, they are created, producing the configuration shown in Figure V-15.2.

The ART "the", the NUMBER "four" and the NOM "subs" are combined by rule NP10 to form the NP "the four subs". The ILR of this phrase (Figure V-14) has taken attribute-value pairs from each of the constituents. The SUPSET, CMU, NBR and NET are taken from the NOM, the DET, for determiner, from the ART, and the NUM, for actual numeric-count, from the NUMBER.

The network representation of the NP, Figure V-15.3, also reflects information taken from each of the NP constituents. From the NOM, the node 'G1' and the s arc to 'SUBS' are taken. Since the numeric-count of the subset size is given by the NUMBER, a card arc is created from 'G1' to '4', indicating that the cardinality of set G1 is 4. Furthermore, by the ART this set is indicated to be a reference to some known set (as opposed to a description of an unfamiliar set), and hence the node 'G1' representing the set is marked by (DET DEF), meaning that it is definitely determined.

The transformation of the DIGIT "one" into a NUMBER parallels the transformation of "four". However, the NUMBER "one" is further transformed into a NUMBERP (which includes such NUMBER-like constructs as "how many" and "more than four").

Rule NP3 is used to combine the NUMBERP "one", the token "of" and the NP "the four subs", into the NP "one of the four subs". The interpretation of this phrase is that attention is being called to some element of the set G1 consisting of "the four subs." (Had the number been "two" rather than "one", attention would be called to a subset with cardinality two.) This interpretation is conveyed by both the ILR and the network. The ILR shows the NBR of the phrase to be S (singular). Furthermore, the supset of the phrase is not some node in the knowledge net (such as 'SUBS'), but rather is the derived construct "the four subs". This difference between a direct and derived supset is indicated by the use of attribute SUPSET* as opposed to SUPSET. The SUPSET* of the NP will be recognized as the ILR of "the four subs".

In terms of the network, the NP is represented as in Figure V-15.4. Node 'G2', with its e arc to 'G1', represents one of the elements of G1, the set constituting "the four subs". Although the network representation of the NP has two nodes in the scratch space, 'G2' may be thought of as the immediate interpretation of the NP, with 'G1' aiding in the definition of G2. The semantic dominance of 'G1' by 'G2' is reflected in the ILR. The NET component of the total NP is 'G2' while 'G1' is the value of the NET attribute of the SUPSET* of the total NP. Since 'G2' is the NET of the top level, it is called the head node of the network representation.

The analysis of the NP "the U.S." parallels the discussion above, but, of course, is much simpler. The network representation of this phrase is shown in Figure V-15.5. "The U.S." is represented by G3, a definitely determined element of USAS, the set of all countries called the "United States". Since the cardinality of USAS is one, the definite determiner will cause G3 to be mapped onto the single element of USAS at evaluation time. That is, 'G3' is to be interpreted as a reference to some node already in the knowledge net. Since there is only one USA in the knowledge net, 'G3' will be associated with that (the only) USA.

The transformation of the V "own" and -SG "s" into a VP is very similar to the transformation of "subs" into a NOMHEAD. The crucial step is performed by rule VP1 which produces the VP. This rule causes a node to be created in the scratch space (see Figure V-15.6) which represents an owning situation, an element of the set OWN.RELS, the set of ownership relations. This linkage to concrete information in the knowledge space is determined solely by the SUPSET attribute.

The ILR of "owns" contains the attributes PDGM (paradigm) and MANDATORY at all stages of its evolution. The value of the PDGM attribute of a verb-like constituent is the name of a short code segment that aids in assigning surface cases (such as subject, direct object, and prepositional phrases) to deep cases (semantic attributes) of the verb-like constituent. For "owns", the PDGM is PG.OWN, the own paradigm. The value of the

MANDATORY attribute is a list of deep cases that must be filled for the verb-like constituent to be complete.

The VP "owns" and the NP "one of the four subs" are combined to form the VP "owns one of the four subs" by rule VP2. This rule assumes that the input NP is to fill one of the deep case arguments of the input VP. To produce a meaningful resultant structure, the rule must determine which deep case the input NP fills, to see if the NP encodes a satisfactory argument for that case, and construct an appropriate network linkage between the VP concept and the NP concept.

The determination of what deep case (if any) the NP fills is aided by the code segment that is the value of the VP's PDGM attribute. This code considers the position of the NP (or, in other instances, the PREPP or S) relative to the verb, the VOICE of the verb, and the deep cases already assigned arguments (and for VP => VP PREPP, the preposition used). PG.OWN, the paradigm code for "own", hypothesizes that an NP to the right of the verb specifies the #@ownee and an NP to the left of the verb specifies the #@owner. If the VOICE of the "own" is PASSIVE, PG.OWN hypothesizes that an NP to the left of the verb specifies the #@ownee and a PREPP with preposition "by" specifies the #@owner. (If VOICE is unknown, the presence of a "by" PREPP satisfying #@owner requirements will cause the VP to be marked as PASSIVE.) For the example at hand, the NP is hypothesized to specify the #@ownee of the owning situation.

Once a hypothesis has been made concerning which deep case the NP fills, a test is conducted to see if the object specified by the NP is semantically qualified to fill the hypothesized case. (If the test fails, a message is sent to the paradigm code and either a new hypothesis is made or the rule fails.) This determination is made by consulting the delineation (definition) of the verb-like component's super category. For our example, the delineation of OWN.RELS, encoded in space -own-, is examined. (See Figure V-10.) The delineation of OWN.RELS indicates that the #@ownee of an owning situation must (for our domain) be an element of PHYSOBJS, the set of physical objects. The immediate meaning of the NP "one of the four subs" is represented in the semantic net by node 'G2', since it is 'G2' that represents the one of the four subs that is being talked about. Hence, the assignment of NP to #@ownee satisfies the semantic requirements of the delineation of OWN.RELS if G2 can be shown to be an element of PHYSOBJS. This deduction turns out to be very easily accomplished in the semantic network. G2 is an element of G1, which is a subset of SUBS. (This information is available from Figure V-15.4.) In turn, SUBS is a subset of PHYSOBJS (as seen in Figure V-10) and thus G2 is an element of PHYSOBJS.

With G2 confirmed as an acceptable #@ownee for the owning situation G4, an arc labeled "ownee" is constructed in the scratch space from 'G4' to 'G2', as shown in Figure V-15.7. 'G4' is considered to be the head node of this structure since the NP (headed by 'G2') is an argument to situation G4. The link between 'G4' and 'G2' is also indicated in the ILR of the VP. The attribute-value pair (NET 'G4') appears in the top level list of attribute-value pairs for the VP. Thus, 'G4' is singled out as the immediate network representation of the VP (with 'G2' and 'G1' serving to help define the meaning of 'G1'). The ILR for VPs with known case arguments includes an attribute-value pair with attribute CASES whose value is a list of pairs of the form

(<case-name> <case-argument>)

For the current example, only the #@ownee of the owning situation is known. Hence, only one pair is on the cases list. The <case-name> of this pair is OWNEE and the <case-argument> is the ILR of "one of the four subs".

The value of the attribute PDGM.MESSAGE of a verb-like constituent in Figure V-14 is a piece of data used to restart the paradigm code--the value of attribute PDGM. Typically, the value of this attribute is a list of assignments of own-type variables.

The last significant transformation in the translation of the example sentence is performed by rule S1 which combines the NP "the US" with the VP "owns one of the four subs" to produce a complete sentence (S). The task performed by rule S1 is almost identical to the task performed by rule VP2 which was just discussed. Using the paradigm code and information in the knowledge net, it is determined that "the US" satisfies the requirements of the OWNER case of the owning situation. With this determination made, an arc labeled "owner" is created from node 'G4' to node 'G3' as shown in Figure V-15.8. The ILR of the S looks very much like the ILR of the VP, but the TYPE has been changed to S and the OWNER construction has been added to the list of cases.

The transformation from S to U performed by rule U1 makes no changes in the representations (network and ILR) of the S, but simply checks to see if the S is capable of 'standing alone' (as opposed to being a subordinate-clause type of S). This check entails testing to see whether the MANDATORY case arguments have been filled.

It is important to note that the semantic network fragment constructed in the scratch space as a result of translating the input utterance, "The US owns one of the four subs,", is structurally identical to the network fragment that would exist (or does exist) in the knowledge space to encode the information conveyed by the sentence. Currently, this scratch space network is the end product of the semantic component. However, programs are currently being designed and written that will act on the structures created in the scratch network. For questions, answers will be retrieved and responses made to the user. For statements, such as the current example, the new information may be absorbed into the knowledge space. For statements that do not involve definitely determined components, this absorption is simply a matter of moving nodes and arcs from the scratch space into the knowledge space. For statements involving determiners, as in the current example, the process becomes slightly more complex. Determined nodes such as 'G1' and 'G3' are assumed (as a precept of the speaker of the input sentence) to be references familiar to the hearer. In terms of our system, to be 'familiar to the hearer' is to be encoded in the knowledge space. Thus nodes such as 'G1' and 'G3' are references to nodes that already exist in the knowledge space. To absorb the input information into its general knowledge, the system must find the knowledge space nodes that are referred to by 'G1' and 'G3' and then interconnect them following the structure of the scratch space. For example, to find a knowledge space node resembling 'G3', the system looks for a node with an e arc to 'USAS'. Since 'THE-US' is the only such node, 'THE-US' is substituted for 'G3' in the absorption process.

b. Example 2

Unlike the sample utterance presented above, almost all inputs collected in our protocol experiments were questions. Thus, as a second example of the translation procedures, consider the utterance

Does the US own one of the four subs?

which is an interrogative variation of the previous declarative statement.

The parse tree for this utterance is shown in Figure V-16. Note that many of the same constructs appear in this tree as appeared in the tree of Figure V-13. Figure V-17 presents ILRs of phrases appearing in this second example that did not appear in the first, and Figure V-18 shows the final network representation of the question.

The current example differs from the first primarily in that its S component is formed from an AUXD, NP, and VP by rule S7 rather than from an NP and VP by rule S1. In terms of constructing a representation of the utterance, this difference is rather small, since the composition semantics of rule S7 actually calls the composition semantics of rule S1 as a subroutine. After S1 constructs the structures discussed previously, rule S7 simply marks the POLARITY (whether the statement is true or false) . being in question. This marking is accomplished by adding the pair (POLARITY ?) to the property list of node 'G4' and to the top level attribute-value pair list of the ILR of the S. The ILR top level list is also set to begin with the entry (E?), meaning that the representation contains an embedded question. The (E?) appears as the first entry so that routines that use the ILR can tell immediately whether an embedded question is present.

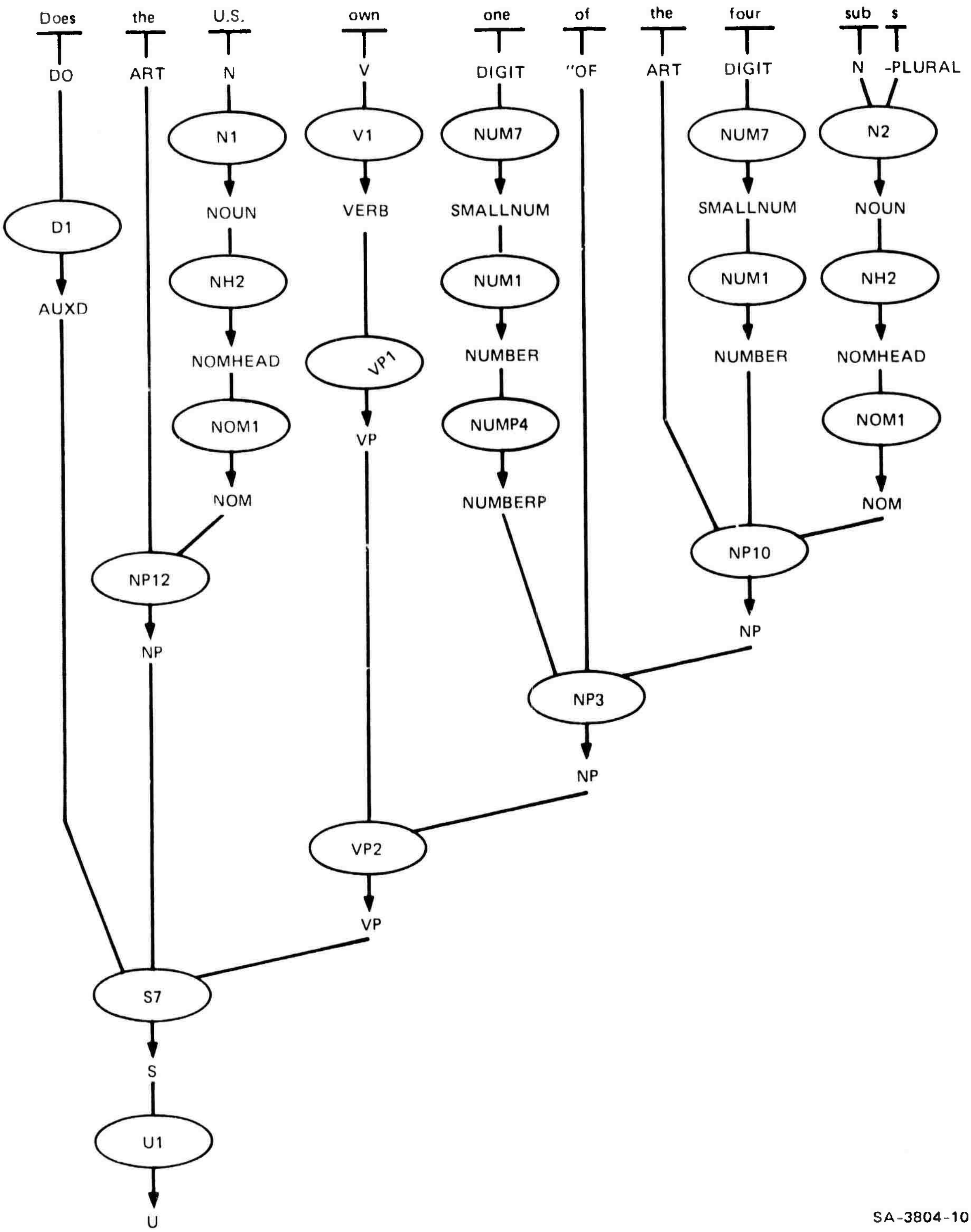


FIGURE V-16 PARSE TREE OF "DOES THE U.S. OWN ONE OF THE FOUR SUBS?"

Figure V-17 Intermediate Language Semantics of Phrases from "Does the U.S. own one of the four subs?"

```
own - VERB
[(TYPE VERB)(SUPSET 'OWN.RELS')(PDGM PG.OWN)
 (MANDATORY (OWNER OWNEE))(NBR PL)]

own - VP
[(TYPE VP)(SUPSET 'OWN.RELS')(PDGM PG.OWN)
 (MANDATORY (OWNER OWNEE))(NBR PL)(NET 'G4')]

own one of the four subs - VP
[(TYPE VP)(SUPSET 'OWN.REL')(PDGM PG.OWN)
 (MANDATORY (OWNER OWNEE))(NBR PL)(NET 'G4')
 (PDGM.MESSAGE NIL)
 (CASES [(OWNEE [(TYPE NP)(NUM 1)(NBR S)(NET 'G2')
                 (SUPSET* [(TYPE NP)(SUPSET 'SUBS')
                           (CMU COUNT)(NBR PL)
                           (NET 'G1')(NUM 4)
                           (DET DEF)])])])]

Does the US own one of the four subs  -  S
[(E?)
 (TYPE S)(POLARITY ?)(SUPSET 'OWN.RELS')(PDGM PG.OWN)
 (MANDATORY (OWNER OWNEE))(NBR S)(NET 'G4')
 (PDGM.MESSAGE NIL)
 (CASES [(OWNER [(TYPE NP)(SUPSET 'USAS')(CMU COUNT)
                 (NBR S)(NUM 1)(NET 'G3')
                 (DET DEF)])
         (OWNEE [(TYPE NP)(NUM 1)(NBR S)(NET 'G2')
                 (SUPSET* [(TYPE NP)(SUPSET 'SUBS')
                           (CMU COUNT)(NBR PL)
                           (NET 'G1')(NUM 4)
                           (DET DEF)])])])]
```

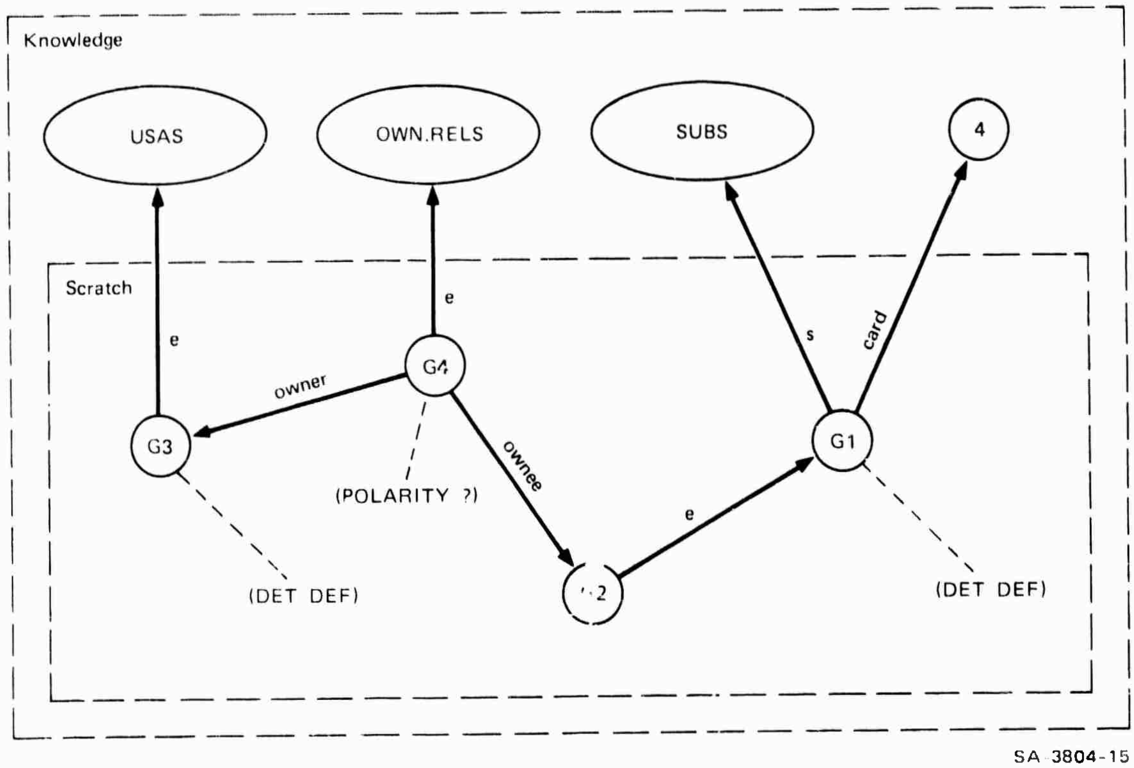


FIGURE V-18 NET SEMANTICS OF PHRASES IN "DOES THE U.S. OWN ONE OF THE FOUR SUBS?"

Again, the network of Figure V-18 is the end product of our current semantic component. However, the structure produced in the scratch space will eventually be matched against information in the knowledge space to determine whether a node exists whose structure matches 'G4'. If such a node is found, the input question may be answered affirmatively.

c. Example 3

To round out the discussion of semantic translation, consider a third example utterance

What is the surface-displacement of the Lafayette?

which will illustrate semantic features not covered by the previous examples. The parse tree, ILR, and network representation of this utterance are presented in Figures V-19, V-20, and V-21, respectively.

The first point of interest in this example is the interpretation of the word "what". In accordance with rule NP8, the DET "what" (as in "what submarine" versus "this submarine") may be transformed into an NP. The ILR of NP "what", as produced by rule NP8 and exhibited in Figure V-20, shows "what" to be three ways ambiguous, having a NBR of either S, PL or M (singular, plural or mass). Only the (NBR S) interpretation is shown in the network of Figure V-21.1. Under this interpretation, all that is known about "what" is that it represents some element of UNIOBJS, the universal set.

The translation of "the Lafayette" parallels the translations of "the four subs" and "the US" which were discussed earlier.

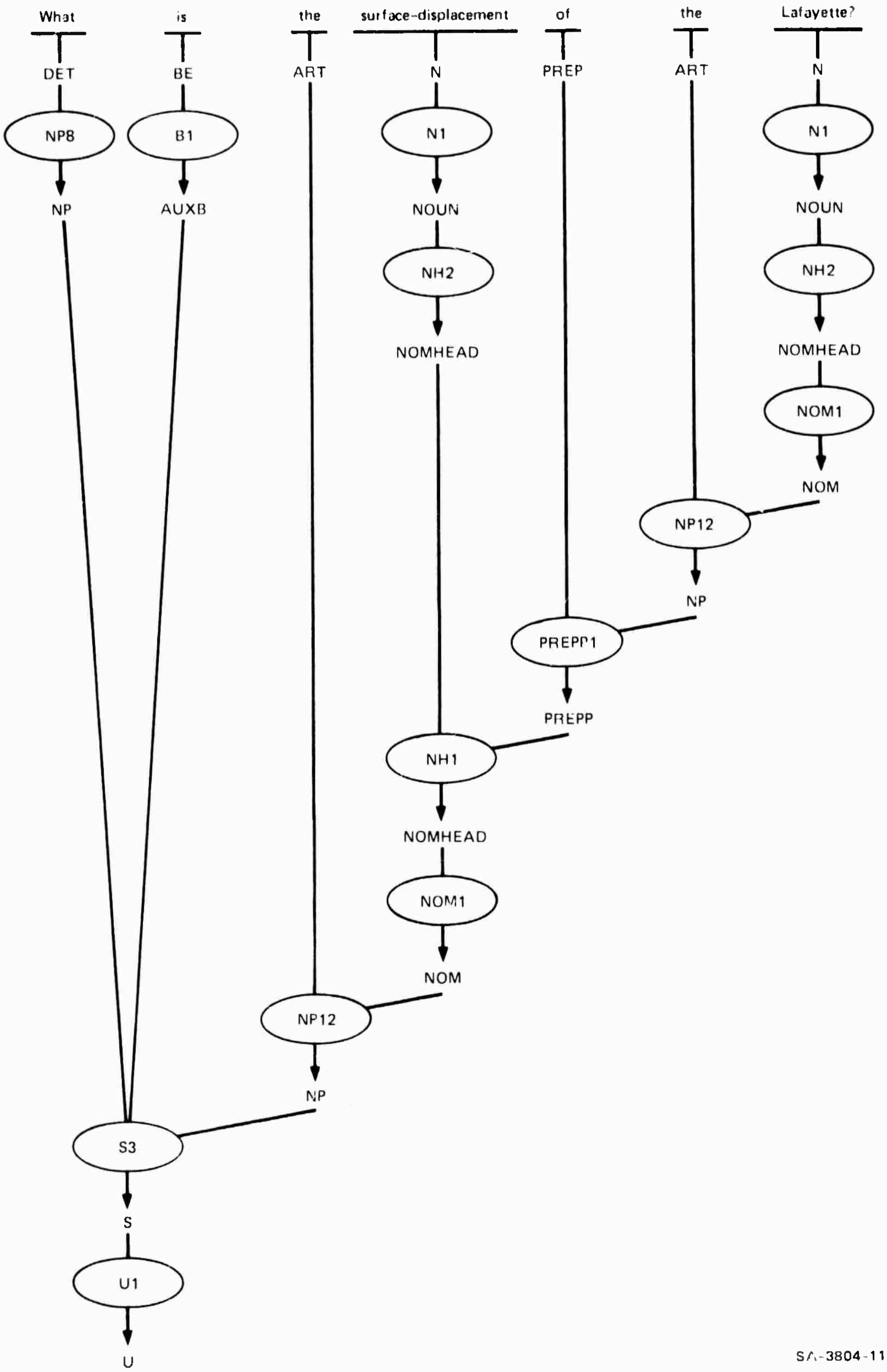


FIGURE V-19 PARSE TREE OF "WHAT IS THE SURFACE-DISPLACEMENT OF THE LAFAYETTE?"

Figure V-20 Intermediate Language Semantics of Phrases from "What is the surface-displacement of the Lafayette?"

```
is - AUXB
[(TYPE AUXB)(NBR (SET M S))]

Lafayette - NOUN
[(TYPE NOUN)(SUPSET 'LAFAYETTES')(CMU COUNT)(NBR S)]

Lafayette - NOMHEAD
[(TYPE NOMHEAD)(SUPSET 'LAFAYETTES')(CMU COUNT)(NBR S)
 (NUM 1)(NET 'G4')]

Lafayette - NOM
[(TYPE NOM)(SUPSET 'LAFAYETTES')(CMU COUNT)(NBR S)
 (NUM 1)(NET 'G4')]

of the Lafayette - PREPP
[(TYPE PREPP)(PREP OF)
 (NP [(TYPE NP)(SUPSET 'LAFAYETTES')
      (CMU COUNT)(NBR S)(NUM 1)
      (NET 'G4')(DET DEF)])]

surface-displacement - NOUN
[(TYPE NOUN)(SUPSET 'SURF.DISPS')(CMU COUNT)(NBR S)
 (INVERTED.HEAD T)
 (INVERSIONS [[(TYPE VP)(SUPSET 'SURF.DISP.RELS')
              (CASES [(MEASURE *)])
              (PDGM PG.BINATT)]])]

surface-displacement - NOMHEAD
[(TYPE NOMHEAD)(SUPSET 'SURF.DISPS')(CMU COUNT)(NBR S)
 (INVERTED.HEAD T)
 (INVERSIONS [[(TYPE VP)(SUPSET 'SURF.DISP.RELS')
              (CASES [(MEASURE *)])
              (PDGM PG.BINATT)(NET 'G2')]])
 (NUM 1)(NET 'G3')]

surface-displacement of the Lafayette - NOMHEAD
[(TYPE NOMHEAD)(SUPSET 'SURF.DISPS')(CMU COUNT)
 (NBR S)(INVERTED.HEAD T)
 (INVERSIONS
  [[(TYPE VP)(SUPSET 'SURF.DISP.RELS')
    (CASES [(OBJECT [(TYPE NP)(SUPSET 'LAFAYETTES')
                     (CMU COUNT)(NBR S)(NUM 1)
                     (NET 'G4')(DET DEF)])
            (MEASURE *)])
    (PDGM PG.BINATT)(NET 'G2')(PDGM.MESSAGE NIL)]])
 (NUM 1)(NET 'G3')]
```

Figure V-20 Intermediate Language Semantics of Phrases from "What is the surface-displacement of the Lafayette?" (continued)

```
surface-displacement of the Lafayette - NOM
[(TYPE NOM)(SUPSET 'SURF,DISPS')(CMU COUNT)
 (NBR S)(INVERTED,HEAD T)
 (INVERSIONS
  [[(TYPE VP)(SUPSET 'SURF,DISP,RELS')
    (CASES [(OBJECT [(TYPE NP)(SUPSET 'LAFAYETTES')
                     (CMU COUNT)(NBR S)(NUM 1)
                     (NET 'G4')(DET DEF)])
            (MEASURE *)])
    (PDGM PG,BINATT)(NET 'G2')(PDGM,MESSAGE NIL)]])
 (NUM 1)(NET 'G3')]

the Lafayette - NP
[(TYPE NP)(SUPSET 'LAFAYETTES')(CMU COUNT)(NBR S)
 (NUM 1)(NET 'G4')(DET DEF)]

the surface-displacement of the Lafayette - NP
[(TYPE NP)(SUPSET 'SURF,DISPS')(CMU COUNT)
 (NBR S)(INVERTED,HEAD T)
 (INVERSIONS
  [[(TYPE VP)(SUPSET 'SURF,DISP,RELS')
    (CASES [(OBJECT [(TYPE NP)(SUPSET 'LAFAYETTES')
                     (CMU COUNT)(NBR S)(NUM 1)
                     (NET 'G4')(DET DEF)])
            (MEASURE *)])
    (PDGM PG,BINATT)(NET 'G2')(PDGM,MESSAGE NIL)]])
 (NUM 1)(NET 'G3')(DET DEF)]

what - NP
[AMBIGUOUS
 [(E?)(TYPE NP)(SUPSET 'UNIOBJS')(NBR S)(ISF ISF)(NUM 1)
  (NET 'G1')(DET ?)]
 [(E?)(TYPE NP)(SUPSET 'UNIOBJS')(NBR PL)(ISF ISF)
  (NET 'G1')(DET ?)]
 [(E?)(TYPE NP)(SUBSET 'UNIOBJS,MASS')(NBR M)
  (ISF ISF)(NET 'G1')(DET ?)]]
```

Figure V-20 Intermediate Language Semantics of Phrases from "What is the surface-displacement of the Lafayette?" (concluded)

```
What is the surface-displacement of the Lafayette? - S
[(E?)
 (TYPE S)(SUPSET 'EQUIV.EXT')(NET 'G5')
 (CASES
  [(E?)
   (THEME1
    [(E?)
     (TYPE NP)(SUPSET 'UNIOBJS')
     (NBR S)(ISF ISF)(NUM 1)
     (NET 'G1')(DET ?)])
   (THEME2
    [(TYPE NP)(SUPSET 'SURF.DISPS')
     (CMU COUNT)(NBR S)
     (INVERTED.HEAD T)
     (INVERSIONS
        [[(TYPE VP)(SUPSET 'SURF.DISP.RELS')
          (CASES
             [(OBJECT [(TYPE NP)(SUPSET 'LAFAYETTES')
                       (CMU COUNT)(NBR S)
                       (NUM 1)(NET 'G4')
                       (DET DEF)])
              (MEASURE *)])
          (PDGM PG.BINATT)(NET 'G2')
          (PDGM.MESSAGE NIL)]])
     (NUM 1)(NET 'G3')(DET DEF)])])]
```

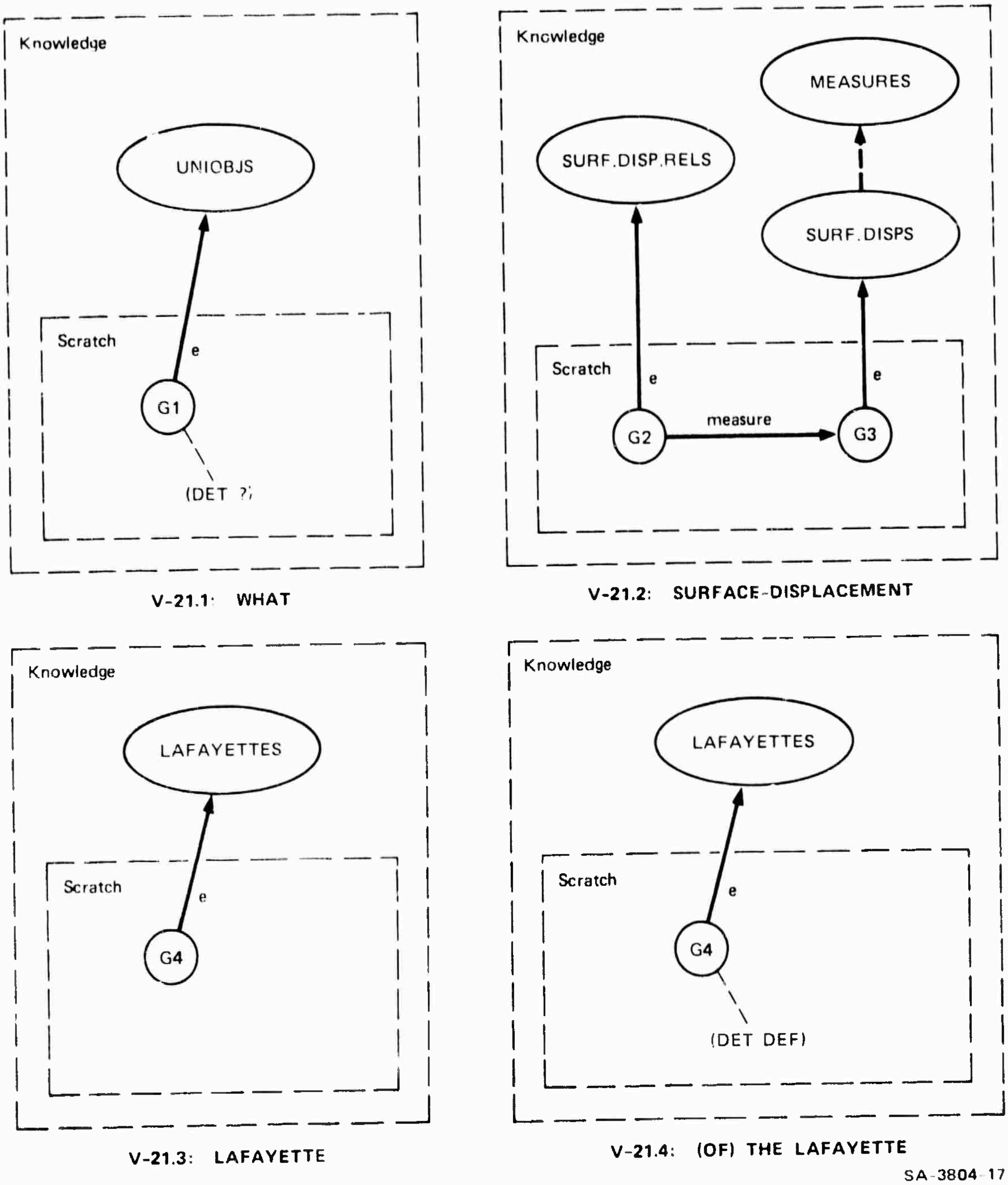
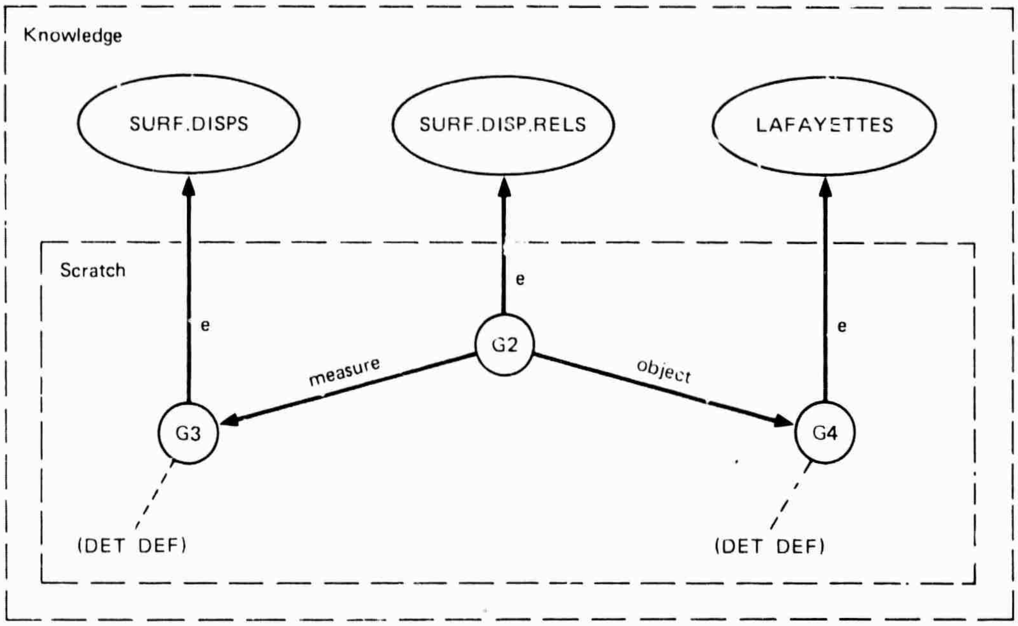
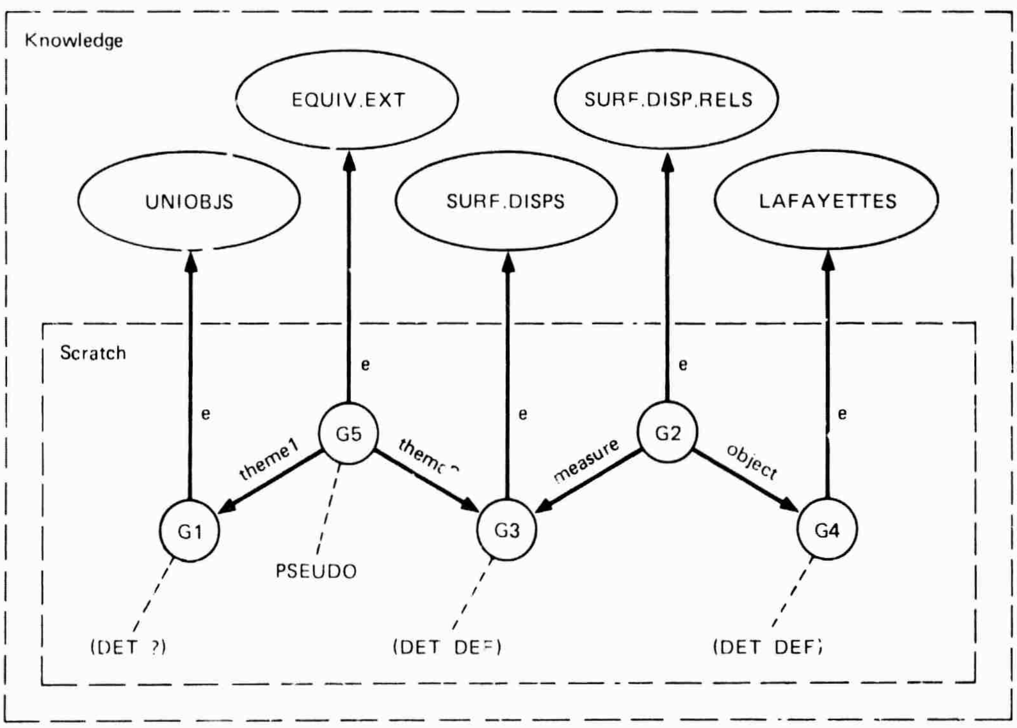


FIGURE V-21 NET SEMANTICS OF PHRASES IN "WHAT IS THE SURFACE-DISPLACEMENT OF THE LAFAYETTE?"



V-21.5: THE SURFA‹ :-DISPLACEMENT OF THE LAFAYETTE



V-21.6: WHAT IS THE SURFACE-DISPLACEMENT OF THE LAFAYETTE?

SA-3804-18

FIGURE V-21 NET SEMANTICS OF PHRASES IN "WHA THE SURFACE-DISPLACEMENT OF THE LAFAYETTE?" (Concluded)

The most interesting point of this third example concerns the translation and interpretation of the word "surface-displacement". ("Surface-displacement" is currently treated as one word, since rules for the treatment of classifiers have not yet been implemented. The reader who so wishes may replace "surface-displacement" with "displacement" or, say, "length".) The word "surface-displacement", unlike words such as "submarine" and "own", carries with it two concepts bundled as one. These two concepts are the concepts of a surface displacement as a weight measure and as a relationship between an object and the weight of water it displaces when floating. Appealing to a more familiar example of such concept bundling, consider the word "owner". An owner is clearly some legal person (a person, corporation, government), but the word "owner" also carries with it the idea that this person is engaged in an ownership relation with some owned object. To say that an owner is simply a legal person is to miss half its meaning. Likewise, to interpret "surface-displacement" as only a weight measure is to miss the relational aspect of its meaning.

The semantic portion of the lexical entry for "surface-displacement", shown in Figure V-12, includes both aspects of meaning. The immediate meaning of this word, reflected by the top level list of attribute-value pairs, is that "surface-displacement" is a singular count noun representing an element of the set SURF.DISPS, a set of weight measures (as seen in the knowledge net of Figure V-10). But this top level list also includes the attribute INVERSIONS whose value is a list of verb-like constructs in which the top-level interpretation (i.e., the weight measure aspect of the total meaning) is a participant. The verb-like constructs on this list are inverted in the sense that verbs typically dominate their arguments as was the case with all VPs discussed in the first two utterance examples. When an ILR contains INVERSIONS, one of the arguments of the verb-like constructs is the principal meaning of the total constituent.

The lexical entry for "surface-displacement" contains only one entry on its list of INVERSIONS, a reference to an element of the set SURF.DISP.RELS, the set of all situations in which an object is related to the weight of water that it displaces while floating. Members of this situation category have two deep case arguments, an #@object (the floating object) and a #@measure (the measure of the weight of water displaced). The ILR of "surface-displacement" makes no mention of the case OBJECT, since its assigned argument is unknown. But the case MEASURE does appear in the CASE list and is shown to have the value *. This * is a pointer to the construction in which the verb-like component is embedded. Thus, the weight measure component of the meaning of "surface-displacement" serves as the value of the MEASURE case argument of the relational component of the meaning.

When rule NH2 is applied to the NOUN "surface-displacement" to produce the corresponding NOMHEAD, dual entries, associated with the two aspects of the NOMHEAD's meaning, are made in the scratch space of the semantic network as shown in Figure V-21.2. The interpretation of this network fragment is that node 'G2' represents a situation (an element of situation set SURF.DISP.RELS) in which some unknown object displaces a volume of water with weight measure G3.

Rule NH1 combines the NOMHEAD "surface-displacement" with the PREPP "of the Lafayette" to produce a new NOMHEAD. The operation of rule NH1 is very similar to the operation of rule VP2 associated with the production VP => VP NP which was discussed earlier. Appealing to the previous discussion, rule NH1 determines that the input NOMHEAD contains an embedded verb-like component in the form of an inversion. A test is then made, using the paradigm code associated with the inversion, to see if the PREPP may fill one of the yet unassigned cases of the embedded verb-like component. For the current example, the PREPP specifies the OBJECT case. The incorporation of the argument carried by the PREPP into the inverted structure is reflected in both the ILR and the network representation (Figure V-21.5) of the resultant NOMHEAD.

Yet another point of interest in the translation of this third example utterance is the application of rule S3 to produce the final S. Rule S3 is used only to combine components of the form

NP AUXB NP.

For the current example, the meaning of the S is "'what' IS-EQUIVALENT-TO 'the surface-displacement of the Lafayette'?" In the knowledge space of the semantic network, equivalent objects are recorded by the same node. Hence, there is no true network counterpart of the relationship IS-EQUIVALENT-TO. To circumvent this difficulty, a node 'G5' is created to encode the equivalence relationship in the scratch net only. Since 'G5' can have no counterpart in the knowledge space, it is marked as a PSEUDO node. Despite the special nature of 'G5', all the usual conventions are followed in its encoding, and the network routines perform in their usual way without considering the PSEUDO property. Thus, by an e arc, 'G5' is associated with the knowledge space (PSEUDO) node 'EQUIV.EXT' which encodes the set of all situations in which a #@theme1 and a #@theme2 in the scratch net are equivalent in extension (i.e., are equivalent when mapped into the knowledge net).

When implemented, the routines that act on the network translation of user inputs will process instances of EQUIV.EXT by mapping the #@theme1 and #@theme2 onto the same node in the knowledge space. This mapping may require the merger of two knowledge space nodes. For example, suppose the input is "8200 tons is the surface-displacement of the Lafayette." Then the knowledge space node representing "8200 tons" is merged with the knowledge space node representing "the surface-displacement of the Lafayette" (provided these two descriptions do not already map onto the same node).

For the example utterance "What is the surface-displacement of the Lafayette?", the node for "what" is merged with the node for "the surface-displacement of the Lafayette", causing the merger product to be flagged with the property (DET ?) which indicates that the information content of the merged node is to be output as an answer to the user's query. (The question marker is removed by the output process.) Generation routines determine that the node may be expressed either as "the surface-displacement of the Lafayette" or as "8200 tons". Since the question was posed in terms of the former, a generation controller selects the latter for output.

In answering questions about "the Lafayette", question answering procedures must determine to which node in the knowledge space "the Lafayette" refers. This task is the responsibility of the discourse analysis routines discussed in Section VI, Discourse Analysis and Pragmatics. Examining the network in Figure V-11, "the Lafayette" might refer to some particular Lafayette that is currently under discussion, such as the Von Steuben. If this is the case, the referent node is 'VON-STEUBEN'. However, "the Lafayette" might refer to the generic, in which case the archetypal element 'EN.LAFE' is appropriate, being associated with information that is general to the category LAFAYETTE. Even to find the "surface-displacement of the Von Steuben", processing may pass through 'EN.LAFE' if surface-displacement information has not been explicitly recorded with 'VON-STEUBEN'.

D. Problems and Plans for Improvements

The implementation and testing of the system described above have shown us both the strong points and the weaknesses of our original designs and have provided insights into how improvements in system performance may be achieved. Two insights gained through the construction effort are particularly important. The first of these is the realization that the intermediate language is really not necessary for discourse analysis as was originally supposed. The discourse procedures that have been developed for our current system extract what information they need directly from the semantic net. Certain information from the associated parse tree appears also to be helpful--and is to be combined with network data in the novel way described below. The second major insight concerns our use of partitioned semantic nets. Having gone through one iteration of partitioning implementation, we now see both better ways to encode the partitioning mechanism and new applications for its use. These innovations will be presented shortly. As is usually the case with regard to running systems, we have found that the semantic composition routines run slower and consume more memory than was hoped. However, these shortcomings in efficiency will be at least partially corrected by curtailing the construction of intermediate language representations and by using a more sophisticated partitioning mechanism.

In our first implementation of partitioned networks, for the system discussed above, both the routines that build and manipulate the network and the network data structures themselves were quite straightforward. While the original system had the virtue of simplicity, it also had a major problem of inefficiency, which at first was thought to be an inherent property of performing translations into nets. The inefficiency arises in the following way. Whenever the acoustics mishears a word or the parser (temporarily) takes the wrong path through the grammar, erroneous network structures are produced. While the construction of numerous erroneous structures must be expected in the process of parsing natural language (especially speech), these spurious structures are particularly costly in networks. The cost arises not so much from the wasted effort of constructing inappropriate structures (which must be done in any system of representation), but because the back-linked nature of networks causes the network representations of phrase constituents to be irrevocably altered when these constituents are interlinked to produce the representation of the complete phrase. Thus, if the network representation of an utterance constituent is erroneously used in forming a spurious phrase, its structure becomes altered, rendering it unusable for incorporation under its correct interpretation (or other spurious interpretations). To prevent the network representations of utterance components from being altered, our original system makes a copy of a representation before the representation is allowed to be altered. This copy includes all information in the scratch space that relates to the constituent. The copying process is costly in terms of both computation time and memory space.

Although the constituent modification problem discussed above was anticipated in our original design, the ratio of spurious constructs to valid constructs was expected to be much lower. When it became apparent that the bulk of the semantic processing effort was being wasted in copying existing structures, the original design was carefully rethought. This exercise led to a solution to the problem that entails a more sophisticated use of net partitioning than originally envisioned. The original design for partitioning implicitly incorporated the assumptions that the hierarchy of net spaces would be strictly tree-like and that an arc would lie either in the space of its to-node or in the space of its from-node. By allowing the hierarchy of spaces to be generalized to a partial ordering and by freeing arcs to lie on spaces that are unconstrained by their associated nodes, a solution to the constituent modification problem was made possible.

This solution is the following. Net fragments representing the most elementary sentence constituents are encoded on separate scratch spaces that are direct descendants of the knowledge space. Lying in sister spaces, these fragments are separated from the knowledge space and from one another by the network partition. When a more complex phrase is to be constructed from a set of subconstituents, a new net space is created that is a descendant of all the spaces encoding the phrase's constituents. (Hence the partial ordering.) While information in this new common descendant space is (as usual) invisible from its parent spaces, all the information in the parents is visible from the descendant. New links (and nodes) uniting the components into a representation of the more complex phrase are encoded in the descendant space, leaving the spaces encoding the constituents unaltered and amenable to incorporation in alternative interpretations.

As an example of the application of this scheme, consider the parsing of the utterance

The-power-plant of the-sub was-built by Westinghouse.

using the simplified grammar

```
R1:      S => NP VP
R2:     NP => NP PREPP
R3:     VP => VP PREPP
R4:  PREPP => PREP NP
```

where

```
     NP => the-power-plant
        => the-sub
        => Westinghouse
     VP => was-built
   PREP => of
        => by
```

The scratch spaces created during the parsing of this utterance are shown in Figure V-22, with each box representing a net space and arrows between spaces indicating the partial ordering.

At the start of processing, the knowledge space is already set up. That is, the system knows about power-plants, have-as-part relationships, submarines, building events, and Westinghouse. On spotting the noun phrase "the-power-plant", the system sets up a space, NP1, below the knowledge space in the partial ordering. Within this space, a structure is constructed representing the meaning of "the-power-plant". Similarly, new spaces are set up to encode the other primitive constituents of the sentence. Through the process of parsing, the parser groups subphrases into ever larger units, calling on the composition semantics routines to aid in the process.

Using rule R4, PREP1 ("by") and NP3 ("Westinghouse") are combined to form PREPP1 ("by Westinghouse"). PREPP1 is allocated its own space, but this space contains no new information. However, when VP1 ("was-built") is combined with PREPP1, the space set aside to encode the resultant VP2 is used to record an arc labeled "builder" from node 'was-built' of space VP1 to node 'Westinghouse' of space NP3. This new arc is visible only from space VP2 (and its descendants) and is not visible from either VP1 or NP3. These latter spaces maintain an appearance of being unaffected by the combining operation.

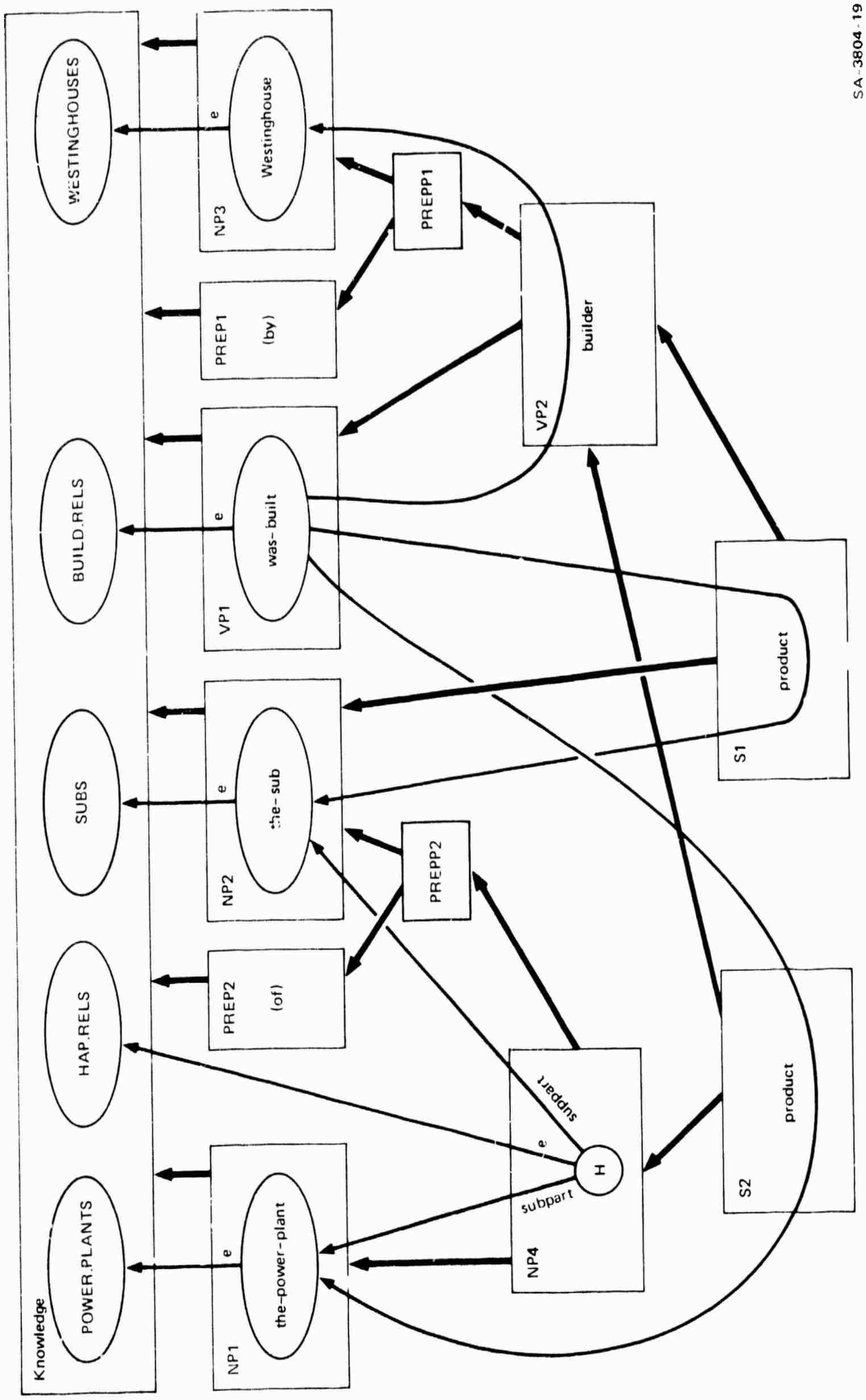


FIGURE V-22 MULTIPLE SCRATCH SPACES FOR “THE-POWER-PLANT OF THE-SUB WAS-BUILT BY WESTINGHOUSE”

Continuing the parse, NP2 ("the-sub") is combined with VP2 ("was-built by Westinghouse") to form S1. The product arc linking the constituent phrases of S1 is contained in space S1 and hence is invisible from the spaces of the constituents. The construct "the-sub was-built by Westinghouse" which is encoded by S1 is a spurious interpretation of utterance components. The reader should note carefully that under our old system the construction of this spurious phrase would alter (and hence, for practical purposes, destroy) both the representation of NP2 and of VP2. As seen below, both these representations are needed in the construction of the correct parse.

Using rule R4, PREP "of" may be combined with NP2 to form PREPP2. The formation of PREPP2 is unaffected by the presence of the product arc from 'was-built' to 'the-sub' which lies in space S1, since all information in S1 is invisible from PREPP2.

Using rule R2, NP1 and PREPP2 may be combined to form NP4 ("the-power-plant of the-sub"). The space encoding this new NP contains a node 'H' and three arcs. While these new constructs are visible from space NP4, they are invisible from constituents NP1 and PREPP2 (and NP2). Furthermore, they cannot be seen from spurious space S1; hence the construction of NP4 has not altered the view of the net from S1.

Using rule R1, S2 is constructed from NP4 and VP2. In addition to the product arc contained in space S2 itself, the view of the net from S2 includes all the information visible from either space NP4 or VP2. This view is summarized in Figure V-23, the view from S1 being depicted in Figure V-24. Since the parse corresponding to space S1 does not successfully account for the fragment "the-power-plant of", it is rejected, and S2 is accepted as expressing the meaning of the input.

The partial ordering of spaces indicated in Figure V-22 is identical to that represented more clearly in Figure V-25. Viewing this ordering from the vantage of space S2 (and ignoring all links to space knowledge) yields the structure of Figure V-26, which, because of the choice of space labels, may be recognized as the parse tree of the input sentence.

The structure thus built by the parser turns out to be well suited for later use by the discourse analysis routines. The semantic representation of the total sentence and each of its syntactic subparts is encoded in a separate net space. Furthermore, the syntactic structure of the input is reflected in the partial ordering of the net spaces as a by-product.

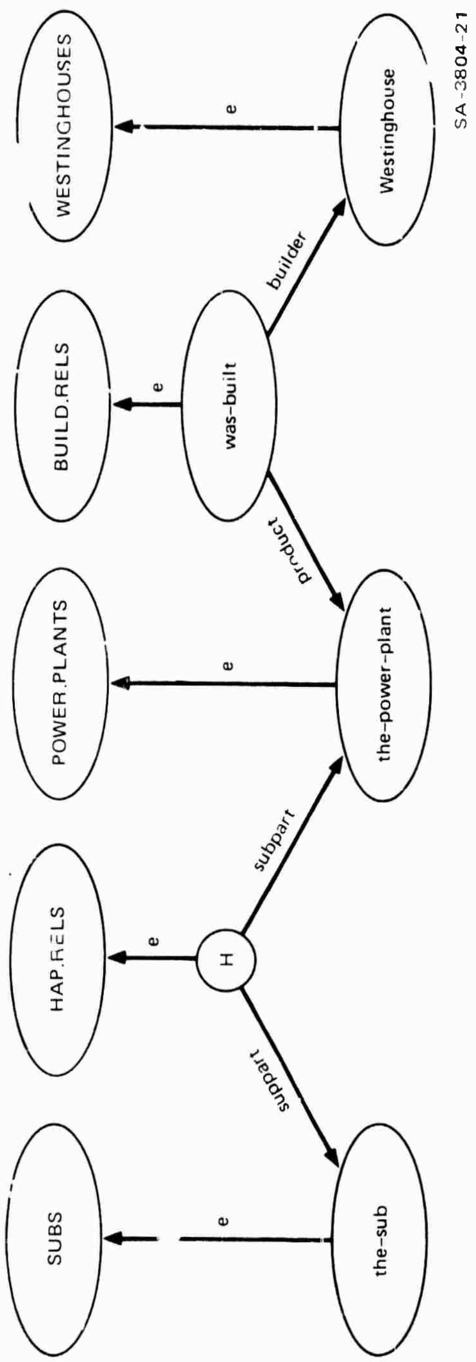


FIGURE V-23 VIEW OF THE PARSE FROM S2

271

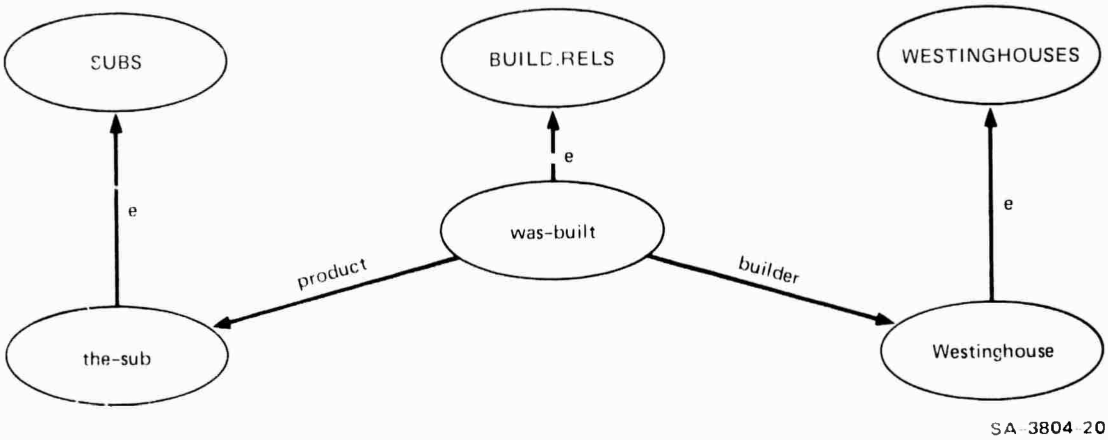


FIGURE V-24 VIEW OF THE PARSE FROM S1

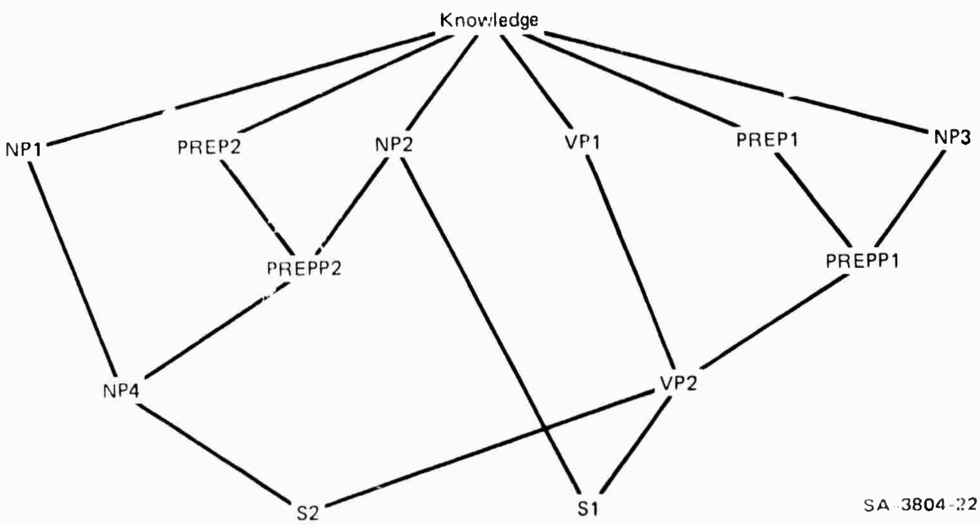


FIGURE V-25 A PARTIAL ORDERING OF NET SPACES

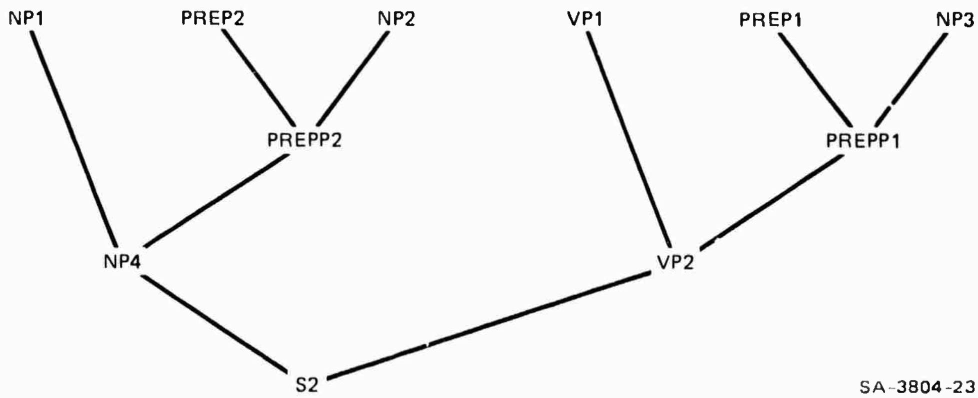


FIGURE V-26 PATHS FROM S2 TO KNOWLEDGE

VI DISCOURSE ANALYSIS AND PRAGMATICS

Prepared by Barbara G. Deutsch

Contents:

A. Introduction

Knowledge about the structure of a task and about the language used by a person in performing that task is essential for the development of a discourse component for the speech understanding system. To get the necessary data, we have been conducting experiments in which we collect protocols from people interacting with simulated systems for both of our task domains. In particular, we are interested in samples of the kinds of language people use when the only constraint placed on them is to restrict the discussion to the given task. This information is needed to determine the subset of English to include in the language definition (see Section IV, The Language Definition). Recordings of spontaneous speech also are necessary for developing and testing the acoustic components of the system. More particularly, for the development of a discourse component, these protocols provide evidence about the relationships between successive utterances in a dialog and between the utterances and elements of the task. Before describing the discourse component we have been developing, it will be helpful to examine these protocols and to consider the different requirements of the two task domains.

1. The Data Management Protocols

The data base used by the System Development Corporation in its previous speech understanding research consisted of a file of the attributes of submarines taken from Jane's Fighting Ships. To obtain natural spontaneous dialogs, we needed to define a set of tasks that would guide people in requesting data. As a first step in defining a set of problems for the subjects to work on, we met with personnel of the Naval Postgraduate School in Monterey and discussed possible applications of our data base, as well as some of the terminology used by people working on submarines. Two kinds of tasks for which the data base might be useful were identified. It could serve as a source of information for people preparing reports concerning the strengths of various submarine fleets, and it could be used by commanders making strategy decisions.

An initial set of experiments was conducted at the Naval Postgraduate School with subjects from the school. The subjects were given a set of charts to fill out and two small problems to solve. The charts were intended to represent ones that might be filled out for a report. The problems were intended to elicit the kind of speech that might occur if the data base were used as an aid in decision making. In addition, the subjects were interviewed after their session both to get more data on terminology and to get feedback on the problems. For the experiments, the data base system was simulated by a Navy officer with relevant experience.

The subjects we used fell, coincidentally, into two categories: those who had experience on submarines but none with computers, and those who had worked with computers but not on a submarine. Of the dialogs we collected, we chose two for intensive study, one representative of each of these classes of subjects. These two dialogs were issued as SUR Note 147. There are interesting differences in the kinds of speech used by the two classes of subjects. One major difference was that the people with computer experience used more stilted language; they specified every parameter of a request completely. This kind of variability may be useful in developing a user model for the system.

These experiments and the interviews that followed them were quite useful in helping us to define an initial set of requirements for the discourse component. However, as a result of our discussions with the subjects, we realized that the problems needed to be made more realistic. We contacted a group at the

Naval Electronics Laboratory Center (NELC) and have defined a new problem: handling a simulated crisis in the Mediterranean concerning the movement of a variety of U.S. and Soviet ships. We will be conducting experiments at NELC soon and expect to use the protocols in further modifications of the system.

2. The Computer Consultant Protocols

In our computer consultant task, an apprentice technician interacts with the system in the maintenance and repair of electromechanical equipment. For our simulations, a person acting as an expert gave advice to the person acting as the apprentice about how to assemble and disassemble an air compressor. The large number of interruptions that occurred in protocols collected when the expert and apprentice spoke directly to each other led us to establish an experimental design in which the two participants were separated and could communicate only through a third person who was responsible for ensuring that the expert and the apprentice did not interrupt one another. SUR Note 146 contains the transcripts of four of the dialogs collected; a description of the experimental design and the facility for gathering data is provided in Deutsch (1974).

The protocols can be divided into those in which the apprentices actually were experienced at working with mechanical equipment and those where they were not. The more experienced apprentices tended to ask questions that were specific ("Do I do X or Y?") whereas naive apprentices asked very general questions

("What should I do next?"). In some of the dialogs it is clear that the (human) expert changes his mode of communicating as he begins to appreciate the skill level of his apprentice. We are examining these data to determine how they can be used in guiding the development of a user model.

One unexpected consequence of the experimental design is that the apprentice may infer that advice is being given to him by a system rather than by a human expert. Thus, we were able to collect some protocols in which the apprentice actually believed he was speaking (albeit indirectly) to a computer. These protocols differ somewhat from the ones in which the apprentice is aware that responses are being generated by another person. For example, in the first case the requests are often more formal although not necessarily in a form that would be easier for the system to process.

3. Discourse Requirements for the Two Task Domains

The discourse component of the current system is capable of handling some of the discourse phenomena that occurred in the protocols we collected for the data management task domain. In the process of implementing these procedures, we have identified better ways of interacting with the semantic component of the system. As a result, we have designed a new framework for the discourse component which should be able to handle dialogs for the computer consultant task domain as well.

There are distinctive differences in the discourse found in the two task domains. A user interacting with the system in the data management task wants to find the answers to questions he has about the information stored in a particular data base. A large number of questions can be asked, and it is not easy to predict which ones would be asked and in what order. That is, although the user obviously has a rationale, it is hard to infer it from his questions. If one could determine what he was 'getting at', then it would be more reasonable to provide him with the information than to make him go through a long series of questions. To build a system that could infer the structure of a question answering dialog would require modeling the intent of the questioner. Since we would want to be able to allow many people to use the same data base for different purposes, it is not clear that it even makes sense to try. In essence, we are not saying that there is no structure to data management dialogs, only that it is hard to determine that structure in a way that is useful for a language understanding system.

In task-oriented dialogs of the kind found in the computer consultant task, the structure of the discourse parallels the structure of the task that is being worked on. Consequently, it is possible to restrict the context that needs to be considered in the analysis of the utterance. Although the particular order of performing tasks is not known, the partial ordering of the subtasks can be encoded, and the small number of topics that are at all likely to to be discussed at any particular time can be determined. It is important to emphasize that this structure is useful precisely because the system can know it a priori.

The two discourse level problems we have been primarily concerned with this year are the resolution of anaphora and the completion of elliptical utterances. An anaphoric expression is one that substitutes for another one, as in the use of pronouns in English to refer to a preceding noun. The identification of that reference requires establishing correspondences with other utterances in the discourse. Elliptical utterances are those with portions missing so that they do not form complete sentences. To identify the missing elements also requires relating them to previous parts of the dialog. In addition to these two major concerns, we have spent some effort studying the use of discourse in a predictive role to anticipate what is likely to be said next.

For all these discourse level problems, in any nontrivial domain, it is necessary for the system to be capable of establishing a local context. By local context we mean the subset of the system's total knowledge base that is relevant at a given point in the dialog. We consider this analogous to determining what is in the focus of attention of the user with whom the system is carrying on the dialog. (It is closely related to the notion of "foreground" developed by Chafe (1972)).

The ability of the system to establish a local context differs for the two task domains. Because of the nature of querying in data management, it is difficult to determine any structure in those dialogs. For this reason, we consider the history of the data management dialogs to be linear, and the focus of attention to be what was said in the previous utterance. (It is clear that what is really needed is to use the previous n utterances for some small value of n.) In the computer consultant domain, however, the structure of the task can be used to establish a local context. Once a focus of attention has been determined, semantic, syntactic, and pragmatic constraints must be used to resolve referances and complete utterances. The procedures we are developing to make effective use of the focus of attention are discussed after a description of our current facility.

B. The Current Capabilities

We first describe how we deal with the limited forms of ellipsis and anaphora occurring in the current set of submarine protocols and then consider extensions needed for handling more general occurrences of these phenomena. We note here that most of the extensions will require including a task model for aiding reference resolution and establishing a focus of attention larger than a single utterance. Also, in the computer consultant task domain, the system's part of the dialog is much more important and will have to be processed more systematically by the discourse routines.

In examining the submarine protocols, we found fairly frequent occurrence of ellipsis (by the professional Navy personnel), but little use of anaphoric reference. Most of the definite noun phrases were generics and there was only one use of a pronoun ("it"). Our initial implementation can process somewhat more sophisticated forms of ellipsis and anaphora than the ones found in these protocols.

1. Ellipsis

We will use the following discourse fragment to illustrate the capabilities of the current system for handling ellipsis. The sequence is typical of the ones found in our protocols.

(1) What is the draft of the Lafayette?

(2) The Ethan Allen?

(3) Submerged displacement?

We initially used the intermediate language representation (ILR) discussed in Section V, Semantics, as the basis for both the ellipsis and the anaphora handling routines. A major reason for using this representation was that it included syntactic as well as semantic information. (An example is discussed below to show the importance of syntactic information.) However, the use of the ILR had several drawbacks. The major problem was that important elements of the sentence often are buried very deep in the ILR structure. For example, Figure VI-1 shows the ILR for utterance 1. The element corresponding to "the Lafayette" is five levels down in the structure! Intuitively, when we consider utterance 1, there are two concepts that seem most important, namely "draft" and "the Lafayette". Any representation that the discourse routines use should make these two concepts stand out. It is clear from the example that the ILR does not have this characteristic.

Figure VI-1 ILR for "What is the draft of the Lafayette?"

```
[(E?)
 (TYPE S)(SUPSET 'EQUIV.EXT')(NET 'G5')
 (CASES
  [(E?)
   (THEME1
    [(E?)
     (TYPE NP)(SUPSET 'UNIOBJS')
     (NBR S)(ISF ISF)(NUM 1)
     (NET 'G1')(DET ?)])
   (THEME2
    [(TYPE NP)(SUPSET 'DRAFTS')
     (CMU COUNT)(NBR S)
     (INVERTED.HEAD T)
     (INVERSIONS
        [[(TYPE VP)(SUPSET 'DRAFT.RELS')
          (CASES
             [(OBJ [(TYPE NP)(SUPSET 'LAFAYETTES')
                    (CMU COUNT)(NBR S)
                    (NUM 1)(NET 'G4')
                    (DET DEF)])
              (MEASURE *)])
          (PDGM PC.BINATT)(NET 'G2')
          (PDGM.MESSAGE NIL)]])
     (NUM 1)(NET 'G3')(DET DEF)])])]
```

In contrast, the semantic net representation for the utterance does emphasize the significant elements. For this reason, the discourse procedures use the semantic net that results from parsing an utterance as the starting point fo. processing.

The semantic net that results from parsing utterance 1 is shown in Figure VI-2. Node N1 represents the fact that the two objects are being equated. In this case, one of the arguments is unknown and to be determined by the retrieval routines. This fact is shown by N2 being an element of UNIOBJS, the set of all objects, without further determination. The property DET that is stored with the node is marked '?'. Nodes N3 and N4 represent number concepts for the value of a draft and for the draft relation, respectively. Node N5 represents an element of the set of Lafayettes. From information stored with the utterance, but separate from the semantic net, the discourse processor can determine that the utterance is a complete sentence. Thus, the only discourse level processing needed is the resolution of anaphora; specifically, it is necessary to determine the specific references for "the draft" and "the Lafayette". The procedures for making this determination are discussed in the section on anaphora below.

When its processing is completed, utterance 1 is added to the discourse history. As mentioned previously, at present the history list contains the sequence of utterances understood up to that time. Each element of the list is the semantic net of the utterance augmented by some syntactic features (e.g., surface subject/object indicators), which the discourse processor uses. When structuring has been added to the history, deep semantic representations will be kept for all utterances ... he surface form will be kept only for the most recent utterance. If this utterance is elliptical, the filled-out version will be kept. In

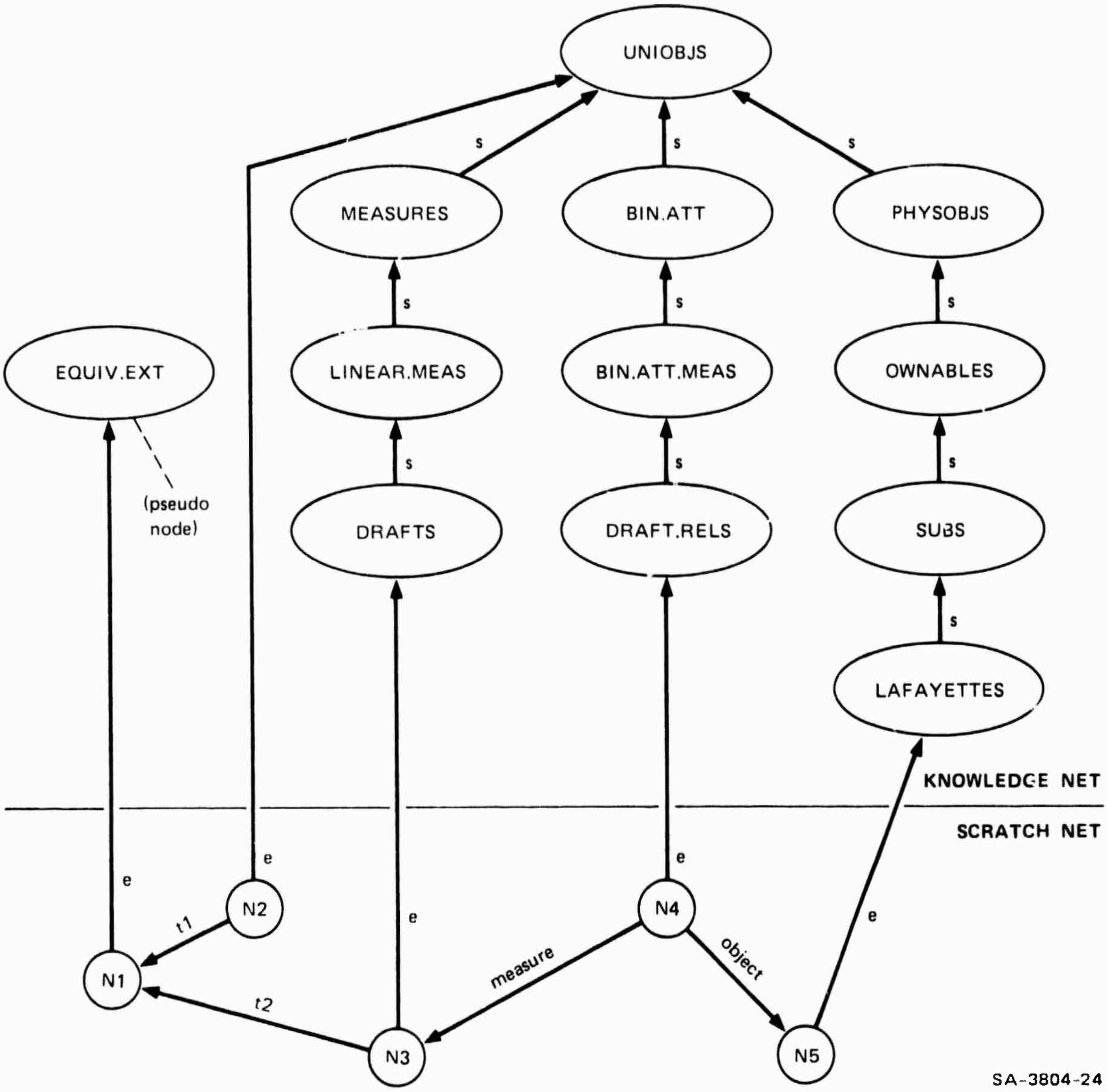


FIGURE VI-2 PARSE LEVEL SEMANTIC NET FOR UTTERANCE 1, "WHAT IS THE DRAFT OF THE LAFAYETTE?"

the new implementation, net space partitioning will be used for recording syntactic features in conjunction with the semantic net, as described in Section V, Semantics.

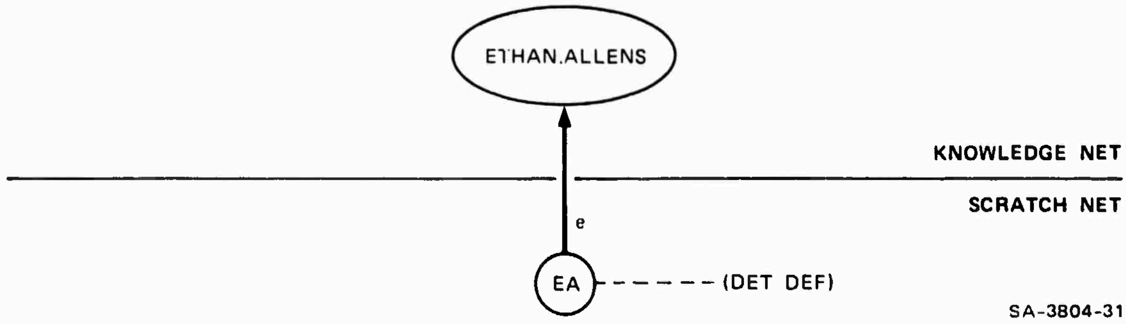


FIGURE VI-3 PARSE LEVEL SEMANTIC NET FOR UTTERANCE 2, "THE ETHAN ALLEN?"

We are now ready to consider the processing of utterance 2. The semantic net that results from analyzing it is a single node in the scratch net, as shown in Figure VI-3. It represents an element of the set ETHAN.ALLENS and is marked as definitely determined. The grammar rule that produced this parse indicates a partial utterance, which must be filled out from the discourse context. Intuitively, we see that the meaning of the phrase is

What is the draft of the Ethan Allen?

That is, the meaning of utterance 2 is equivalent to the meaning of utterance 1 with "Lafayette" replaced by "Ethan Allen". The discourse routines have two problems to solve in reaching this interpretation. First, it is necessary to detect that Ethan Allen matches Lafayette in the previous utterance. Second, it is necessary to determine how much of the structure of utterance 1 should be carried over in expanding utterance 2. When we consider utterance 3, we will see that this last problem is nontrivial. We note that it is clear that any ellipsis must be patterned on the immediately preceding completed utterance. (If it were patterned on an utterance before that, the syntactic patterns of the intervening utterances would interfere.)

We proceed as follows. The last utterance processed, in this case utterance 1, is taken from the discourse history. We want to determine which element of the net corresponding to this utterance is most closely related to the main concept of the new (and ellided) utterance. That is, we want to find what slot in the old utterance the new utterance fills. We use the superset hierarchy of the semantic net for this purpose. The two nodes that are most closely related are so because they belong to a common set that does not include any of the other nodes. That is, considering element (e) and subset/superset (s) arcs, the two most closely related nodes are the ones that have the closest common ancestor. For example, consider the net fragment in Figure VI-4. The sets ETHAN.ALLENS and LAFAYETTES are 'closer' than the sets ETHAN.ALLENS and TORPEDO.TUBES, since it takes two links to find a common superset (OWNABLES) for ETHAN.ALLENS and TORPEDO.TUBES, but only one link to find a common superset (SUBS) for ETHAN.ALLENS and LAFAYETTES.

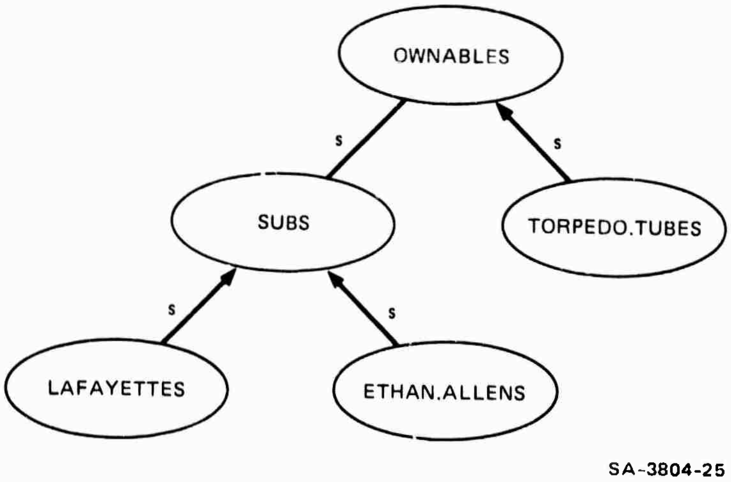


FIGURE VI-4 SEMANTIC NET HIERARCHY

To find the node of the old utterance that shares the closest common ancestor with the new utterance head node, we grow paths along e and s arcs from all the nodes of the old utterance and from the head node of the new utterance. When paths from two different starting nodes reach a common node, it indicates that the two original nodes are elements of a common superset. If one of these two nodes is the head node of the new utterance, the desired match has been found. Note that all paths will eventually reach UNIOBJS. For this reason, any path that reaches UNIOBJS is eliminated immediately. (We also eliminate any node connected to the pseudo-node EQUIV.EXT, because that node only establishes the equivalence of the two structures attached to it.)

The paths traced in our example are shown in

Figure VI-5. Paths from the old utterance nodes are shown with dotted lines; the path from "the Ethan Allen" is shown with a dashed line. In the first step of the application of the algorithm, the paths out of N10 and N11 are eliminated. The new node set is DRAFTS (from N12), DRAFT.RELS (from N13), LAFAYETTES (from N14), and ETHAN.ALLENS (from N15). On the second application of the algorithm, the paths from DRAFTS and DRAFT.RELS are extended to LINEAR.MEAS and BIN.ATT.MEAS, respectively. The paths from ETHAN.ALLENS and LAFAYETTES meet at SUBS. The desired match has been found.

In this example, the merging of the appropriate parts of the new and old utterances is trivial. All that needs to be done is to replace the matching node (in this case N14) with the new utterance node (N15). The fact that replacement can be complicated is illustrated by the case of utterance 3. The parse level net for utterance 3 is shown in the scratch net portion of Figure VI-6; note that it has two nodes. The head concept (determined by semantic routines; see Section V, Semantics) is SDR; it is an element of the set of SUBM.DISP.RELS. The initial matching proceeds as for utterance 2. The result of the path growing algorithm is shown in Figure VI-7. Merely replacing N23 by N25 would give a meaningless structure. In fact, to get the desired interpretation of utterance 3, a whole subnet of the net for utterance 1 needs to be replaced: N23-measure-N22 by

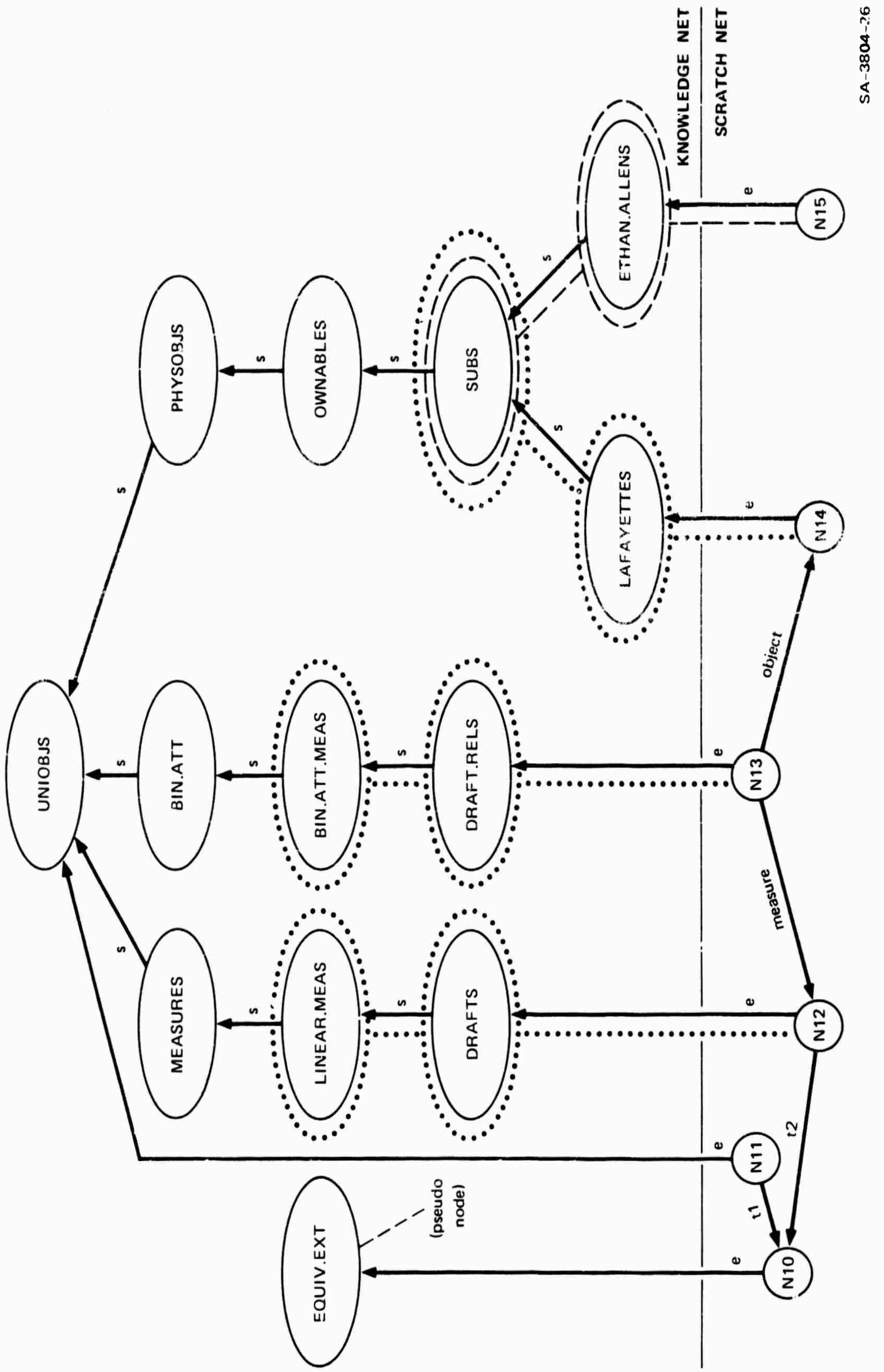


FIGURE VI-5 RESULT OF PATH GROWING ALGORITHM APPLIED TO UTTERANCES 1 AND 2

N25-measure-N26. The utterance expansion routines actually build a new net around the new (partial) utterance using the information from the old utterance net which is not superseded by information in the new utterance.

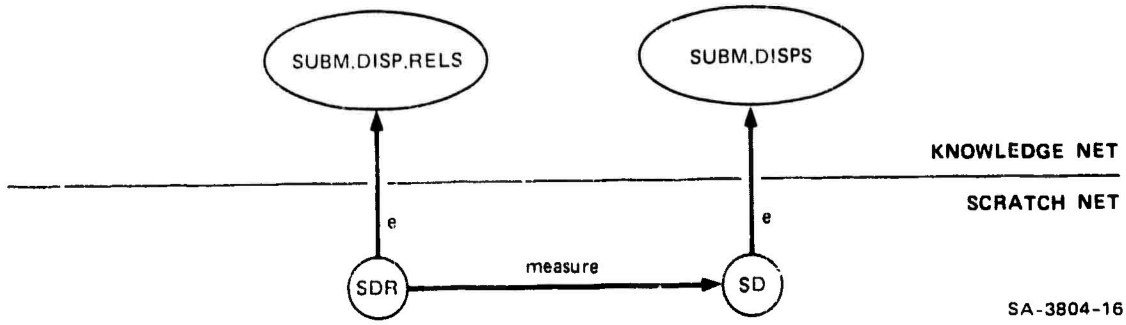


FIGURE VI-6 PARSE LEVEL NET FOR UTTERANCE 3, "SUBMERGED DISPLACEMENT?"

In the above discussion, we assumed that there would be a unique first match. Unfortunately, that is not always the case. It is possible for two nodes in the old utterance to be elements of sets, one of which is a subset of the other. This happens, for example, in

Is the Lafayette a U.S. sub?

More often, it may be the case that two of the elements of the old utterance are members of a common set and hence two paths merge with the new utterance at the same time. This is most likely to happen with comparatives. Consider the question

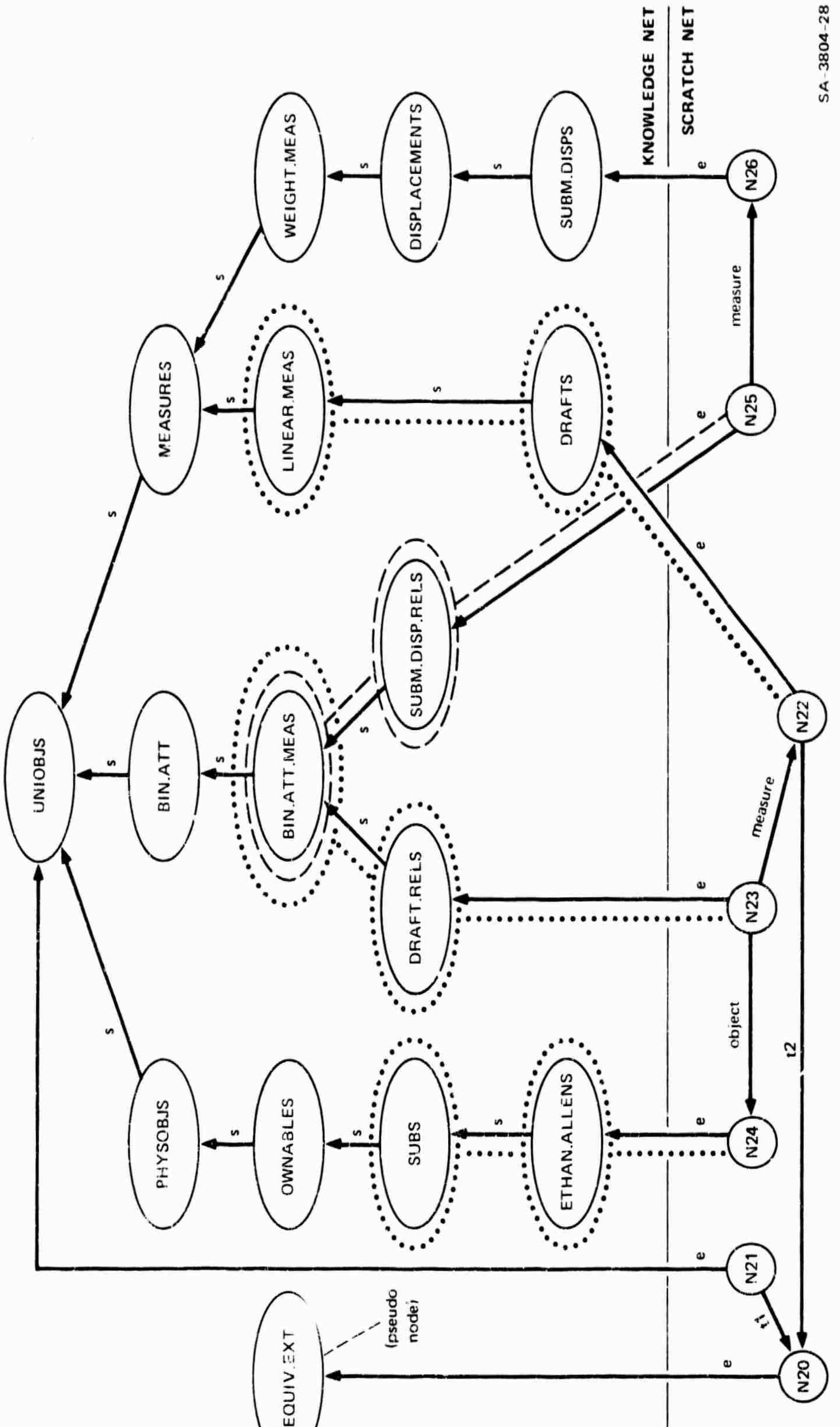


FIGURE VI-7 RESULT OF PATH GROWING ALGORITHM APPLIED TO UTTERANCES 2 AND 3

Is the Lafayette longer than the Ethan Allen?

(Although this question cannot yet be handled by our language definition, it is included because it is a clear example of a discourse level problem.) "Lafayette" and "Ethan Allen" are both members of the class SUBS. Consider what happens if the next utterance is

The George Washington?

Paths from both the node corresponding to "Lafayette" and the one corresponding to "Ethan Allen" will meet at the same time with the path from "George Washington". In this case, there is no further information in the new utterance and the discourse routines will report an unresolvable ambiguity. However, if the second utterance had been either

Is the George Washington?

or

Than the George Washington?

there would have been some syntactic information to disambiguate the semantic match. Thus, the discourse routines use the syntactic markers now kept with an utterance to see if syntactic position can be used to disambiguate.

2. Anaphoric Reference

a. Pronoun References

To resolve pronoun references ("it", "they"), we look at the immediately preceding utterance in the discourse history. For the linear history we are restricted to with the submarine domain, this short perspective is sufficient. When we augment the current discourse capabilities with a structured history (necessary for the computer consultant task domain), we will also add the ability to look back more than one utterance for pronoun references. Note that in looking at more than one utterance back we will be looking up the structure, not linearly back (cf. Deutsch, 1974).

The basic strategy we follow is to look at the case slots that the pronoun fills and find the restrictions on those slots. Then the previous utterance is searched for a concept that satisfies those restrictions. Consider the following sequence:

(4) What is the length of the Ethan Allen?

(5) What is its speed?

The restriction on the antecedent for "it" is that it be an entity that can have a speed. That is, it must be some physical object capable of motion; in the submarine domain, that means a submarine. The only submarine mentioned in the local discourse is the Ethan Allen. And, intuitively, it is clear that utterance 5 is really requesting the speed of the Ethan Allen.

To see how the program reaches this conclusion, we need to look at the semantic net structure resulting from parsing these utterances which is shown in Figure VI-8. "It" is represented by node N34 which is an element of UNIOBJS and (not shown in the net, but recorded with the utterance) is definitely determined and singular. To determine the case slots filled by "it", we look at all of the arcs coming into N34. The only arc to node N34 is the object arc from N33, which is an element of SPEED.RELS. To determine the restrictions imposed on this case slot, we need to look at the delineating element for SPEED.RELS, shown in Figure VI-9. Note that this full description is not stored explicitly in the semantic network, but is implicit from the network hierarchy. That is, the description is built from the delineating elements for SPEED.RELS and from the concepts that are supersets of it in the network. (See the discussion in Section V, Semantics, for further elaboration.) In this case only BIN.ATT.MEAS is relevant. The restriction on the item filling the case argument 'object' is that it be a member of the class PHYSOBJS. Note that if there were more than one case slot (e.g., in conjunction, "the x and y of it"), the restrictions would be the union of the individual restrictions.

To find the elements of the preceding utterance that fit the restrictions, we follow element and superset arcs from the nodes of the utterance in a manner similar to the one used for matching elements when an ellipsis occurs. In the case of ellipsis we have a filler for a slot, but need to determine

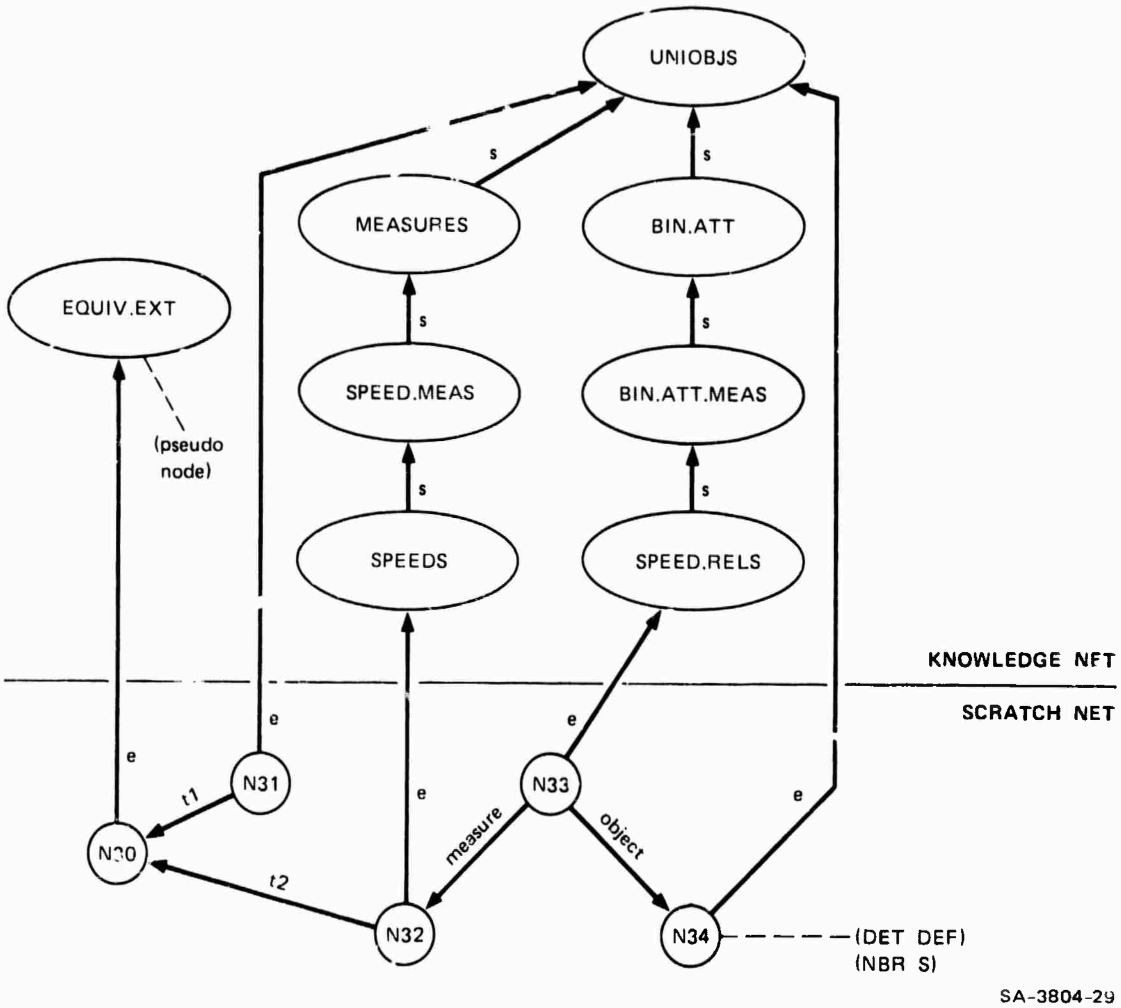


FIGURE VI-8 PARSE LEVEL SEMANTIC NET FOR UTTERANCE 5, "WHAT IS ITS SPEED?"

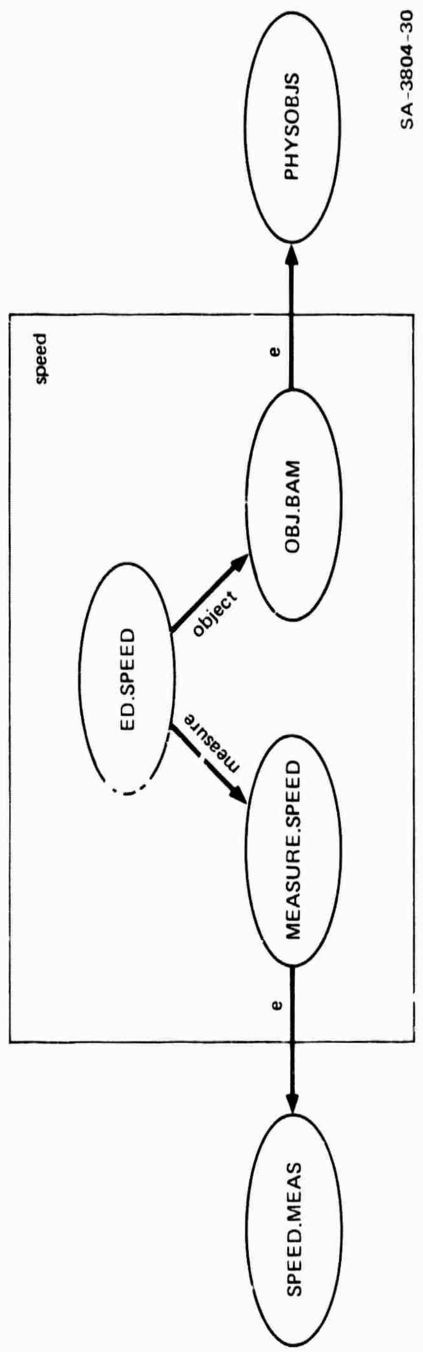


FIGURE VI-9 EXPANDED DELINEATION ELEMENT FOR SPEED.RELS what the slot is; in the case of a pronoun reference, we know the slot (it is filled by the pronoun) but need to find something specific with which to fill it. The node corresponding to the restriction class (in the example PHYSOBJS) is marked, and, using the path growing algorithm described above, paths are grown from all the nodes of the old utterance until an intersection with the restriction class occurs. At this point we have a semantic match. Before the match is accepted and the replacement made, syntactic agreement checks (e.g., for number and, where appropriate, gender) are made.

Again, there may be an ambiguity: more than one node in the old utterance may match at the same time (i.e., on the same step of the algorithm). In this case, all matches are considered as candidates, and factors such as syntactic position are used to find the best match.

In essence, then, the resolution of anaphoric reference is done primarily on semantic grounds. Other factors are considered only when semantics is not sufficient for determining a unique antecedent.

b. Definite Noun Phrases

The network matching that has to be done to resolve definite noun phrases is a subset of what must be done to answer questions. This may be seen by considering the phrase

The U.S. submarine

The parse level network for this phrase is shown in Figure VI-10. Another interpretation of this structure is

The submarine owned by the U.S.

In fact, this structure is exactly the one that would have to be matched to answer the question

Which submarine is owned by the U.S.?

A general package of network matching routines is being written to service both the needs of the question answerer and the discourse routines. At this time, the definite noun phrase resolver is not implemented.

3. Limitations of the Local Routines

There are several limitations to the current discourse package, caused primarily by our dependence on a linear history. The implementation of multiple partitionings in the semantic network and the addition of a focus space partitioning (discussed in the next sections) are aimed at overcoming some of these problems. We discuss other limitations of the current system here.

First, the current implementation depends on being in a question answering environment. It assumes that the syntactic structure of the system's answer to a user's question is not

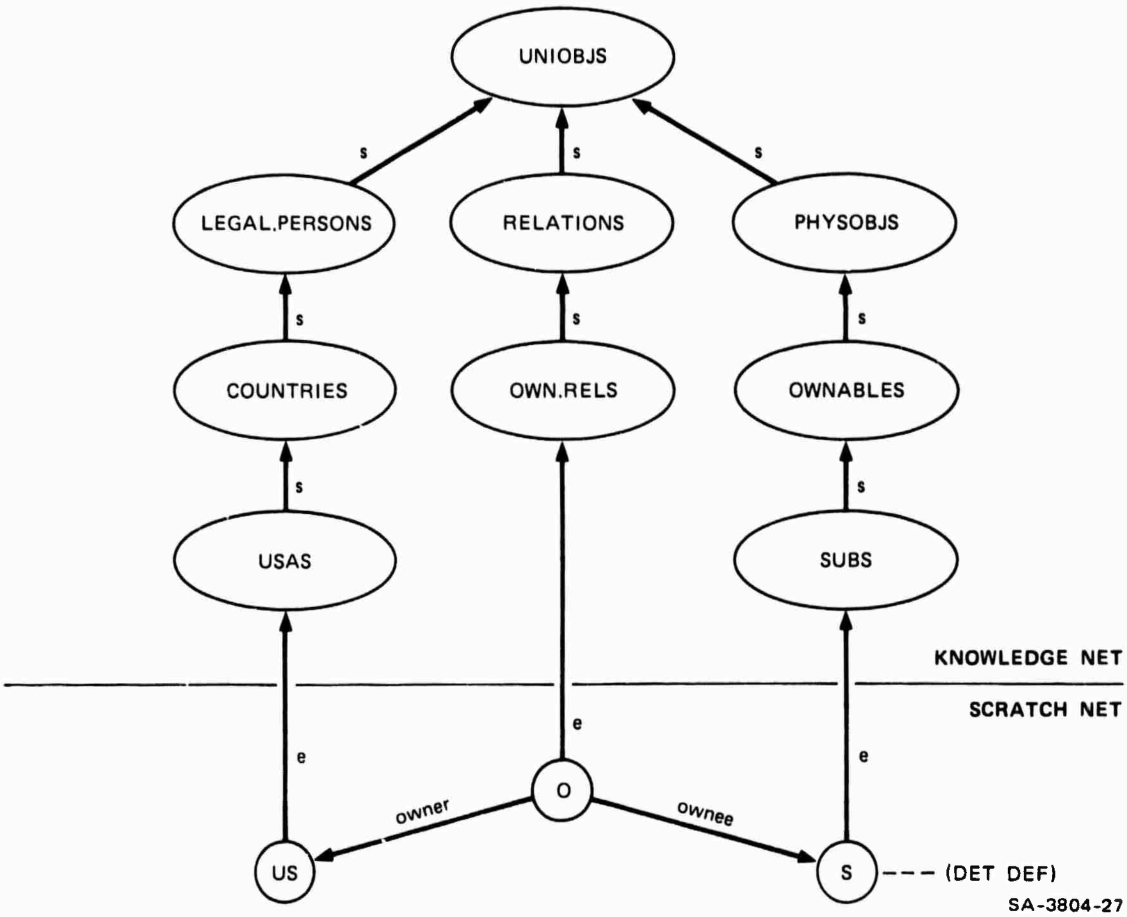


FIGURE VI-10 PARSE LEVEL STRUCTURE FOR "THE U.S. SUBMARINE" relevant to the structure of the next question. For the computer consultant task domain, this assumption is not valid. In that domain, the user and the system are carrying out a true dialog. Both the questions and the answers are important to the dialog history. For this reason, the dialog history will include both system and user generated utterances and will keep track of question/answer parity. As an example, consider the sequence

SYSTEM: Which bolts are tightened down?

USER: The front ones.

To understand this response, first "ones" must be resolved to "bolts", then the NP, "the front bolts", must be used to replace the NP, "which bolts", in the question. These steps require matching the two concepts and then replacing the complete NP.

A second (and related) limitation of the current system is the relatively small use of syntactic information in resolving references. The use of net space partitioning to encode the parse tree will help here. One major concern is being able to determine the scope of noun phrases. Consider the sequence

What is the draft of the diesel sub?

The nuclear sub?

The networks for these two utterances are shown in Figure VI-11. (The representation of fuel-type in Figure VI-11 is a shorthand to simplify the net for illustrative purposes.) In expanding the second utterance, it is necessary to be able to pick up the part

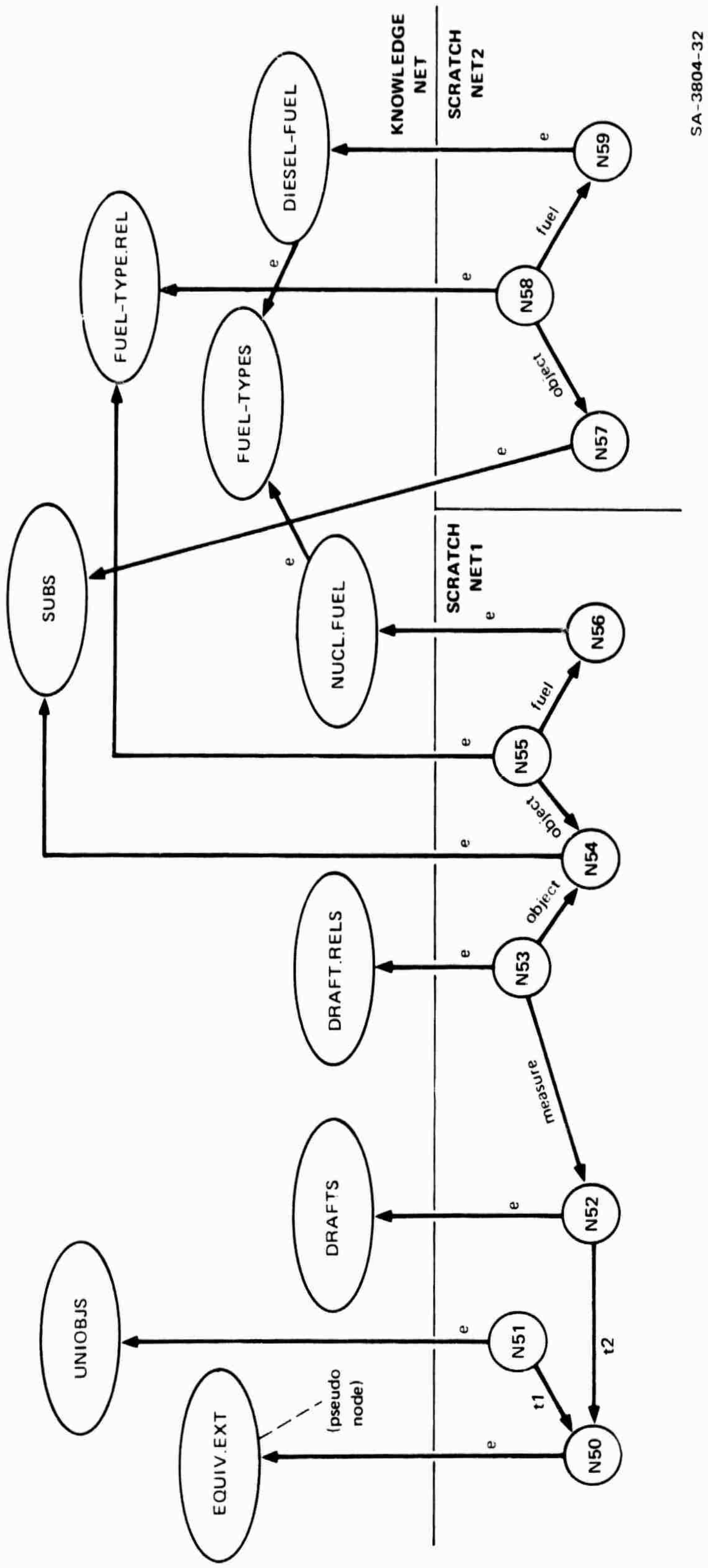


FIGURE VI-11 SEMANTIC NETWORKS FOR THE PARSE LEVEL OF
(NET1): "WHAT IS THE DRAFT OF THE NUCLEAR SUB?"
(NET2): "THE DIESEL SUB?"

301

of net1 corresponding to "draft", but not to pick up the part corresponding to "diesel". This is clear syntactically from the fact that "diesel" is part of the noun phrase with sub, but "draft" is not.

Finally, the discourse routines currently work on a complete utterance. They need to be modified to work on parts of utterances and to return to the parser knowledge about missing information that must be provided if the utterance is to be understood. This knowledge would be used to guide predictions and to influence scoring and other evaluation procedures.

C. The Need for Attention Focusing

The need for an ability to establish a focus of attention can be seen most clearly in terms of the computer consultant task domain. There, not only reference resolution but also the generation of object and action descriptions require that the system have the ability to restrict its attention to a small but pertinent subset of its total knowledge.

In task-oriented dialogs, the dialog context is actually a composite of three different component contexts: a verbal context, a task context, and a context of general world knowledge. The verbal context includes the history of preceding utterances, their syntactic form, the objects and actions discussed in them, and the particular words used. The task context is the focus supplied by the task being worked on. It includes such information as: where the current subtask fits in the overall plan, what its subtasks are, what actions are likely to follow, what objects are important. The context of general world knowledge is the information that reflects a background understanding of the properties and interrelations of objects and actions; for example, the fact that tool boxes typically contain tools and that attaching entails some kind of fastening.

An important aspect of the reference problem is determining what sources of knowledge should be accessed to resolve a reference. Decisions must be made concerning how much effort should be spent testing one antecedent candidate and how much effort should be spent investigating the different context perspectives from which that candidate may be viewed. To illustrate this point, consider the question

Where are the setscrews?

in the context of a preceding command

Tighten the setscrews with an allen wrench.

The phrase, "the setscrews", in the question, must be resolved as the one previously mentioned in the command. This resolution comes from the verbal context (or dialog history). However, in the context of the command

Attach the pump pulley next.

the question can be understood only if the consultant is aware that installing and tightening some screws are part of the operation of attaching the pump pulley. The resolution comes from knowledge of the task. Any screws mentioned in the previous dialog would probably be irrelevant. Finally, if we consider as context the statement

I have the parts box.

then the reference can be resolved by knowing that screws are typically stored in a parts box.

The reference resolver must consider as candidates for antecedent not only objects and actions that are explicitly represented in the dialog history (which would work only for the first of our examples) but also the interconnections of those objects and actions in the task domain and in general world knowledge. It is necessary to decide which kinds of connections to consider first and how long to investigate them before looking at others. It also is necessary to determine how much effort should be put into looking at all the connections of one object or action before considering others. The implementation of the reference resolver is being designed so that we can experiment easily with different strategies for looking at the various contexts. The separation between the three context components is made explicit. The task context is supplied by a connection to a model of the task. The difference between the local verbal context and general world knowledge connections is reflected in the way the semantic representation of the discourse history is kept. Essentially, the local context is separated from global knowledge, but a few links are maintained, as described below.

The problem of object description is closely related to the reference problem, essentially as its inverse. An object is unambiguously described if the description given can be used to locate it uniquely. Any object has a multitude of attributes; some are simple (e.g., color, shape), and others involve connections to other objects (e.g., on-top-of, inside). However, at any one time only a few of these properties are needed to specify an object uniquely, because context limits the other objects from which it needs to be distinguished. As an example, consider the situation when the apprentice is using a 1/2" box-end wrench and a 1/2" socket wrench to tighten a nut/bolt fastening. The two wrenches can be distinguished by type: "the wrench" is ambiguous, but both "the box-end wrench" and "the socket wrench" are unambiguous. However, if the apprentice is using two 1/2" box-end wrenches for his task, they need to be distinguished by other criteria, such as which is on the nut.

D. Focus Space Partitioning

Since the system's knowledge is recorded in a semantic network, a form of net partitioning may be used to group together the facts that are likely to be pertinent at a given point in the dialog. (See Hendrix, 1975, and Section V, Semantics, for detailed descriptions of net partitioning.) For task-oriented dialogs, the division of the dialog into cohesive subdialogs is closely tied to the task structure. In our system, the structure is embodied in the procedural net, which encodes the task structure in a hierarchy of subtasks and allows the representation of partial ordering of steps (Sacerdoti, 1975; Nilsson et al., 1975). By grouping the information relevant to each subtask into a separate net space and ordering the net spaces in accordance with the procedural net hierarchy, a knowledge structure is produced that supplies contextual focus.

Figure VI-12 is the semantic net representation of a wrench W and its relationships to other objects (by 'objects' we mean any entity that is encoded as a node in the semantic net). Note that this is a fragment of a larger semantic net; only a subset of the relationships in which W might participate is indicated. The partitioning shown is the logical partitioning described in Section V, Semantics. Space Sw of this partitioning is used to delineate the class of wrenches, and indicates that each wrench has a size and an endtype. These components of a wrench's description are indicated through case relationships because they are time invariant, intrinsic properties. Neither the size nor the endtype of a wrench may be altered without destroying the wrench itself. Node structures could have been used to encode this information but such an encoding is more expensive and is not needed here. Wrench W, an element of WRENCHES, has size 1/2" and

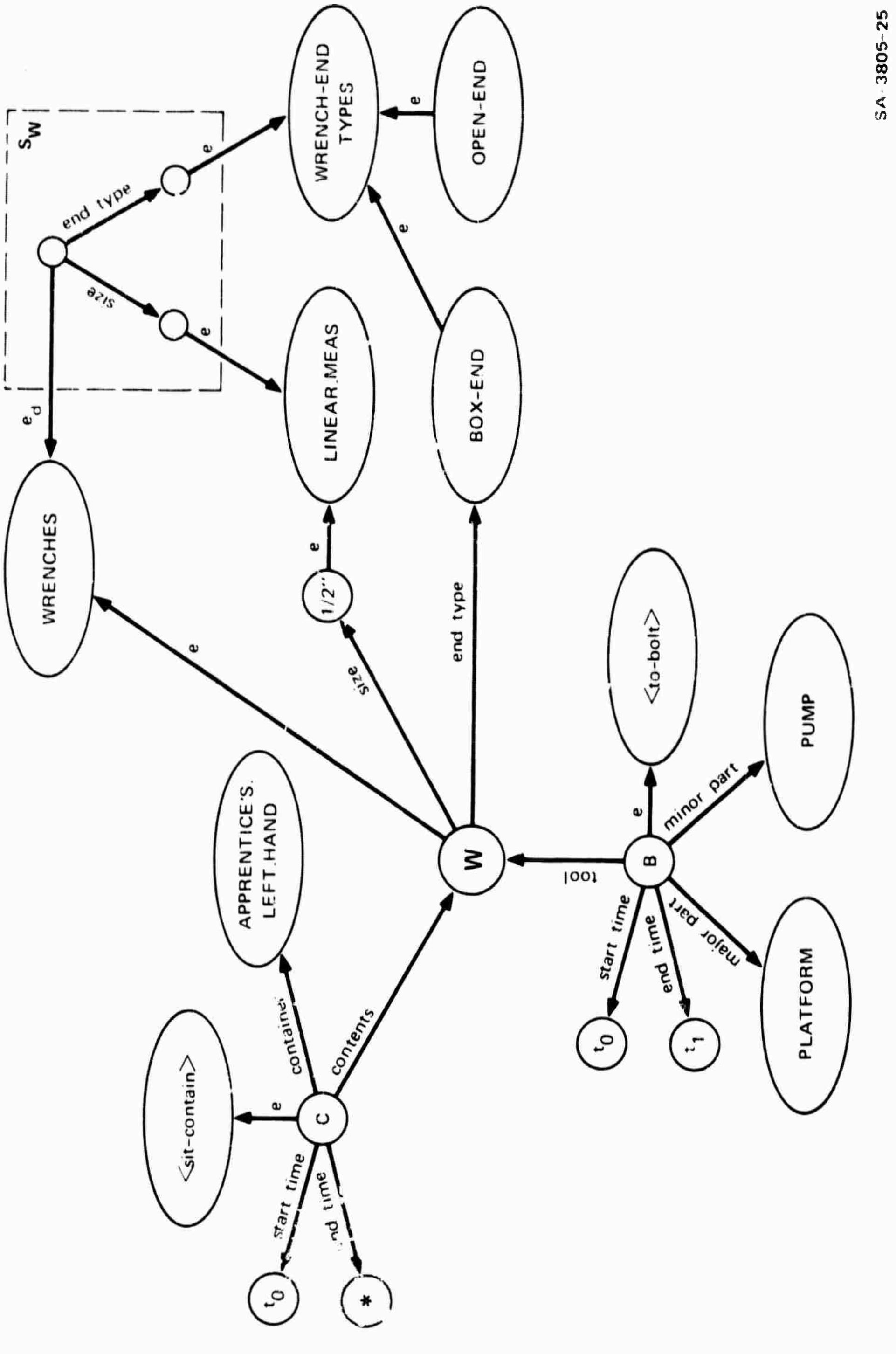


FIGURE VI-12 SEMANTIC NET FOR WRENCH, W, WITH LOGIC PARTITIONING endtype BOX-END. In addition to the intrinsic properties of size and endtype, wrench W has the distinction of having been used (as the tool) in the attaching of the pump to the platform between times Ti and Tj and of being in (being the content of) the apprentice's left hand from time Tk to the present. (Note that "time" arcs go to intervals. An entity also may have a start-time and a end-time; in this case the interval is (endtime,starttime).)

All this information is part of the history of wrench W. As such, any of it may be used in the description of W. However, in any given contextual focus, only some of it is valuable. For this reason we would like to be able to highlight certain arcs and nodes in the network while they are in focus, letting them return to their unhighlighted state when the focus changes.

To do this, network partitioning is used in a new way. Nodes and arcs belong to both logical and focus spaces. The logical and focus partitions are orthogonal to one another in the sense that the logical space on which a node or arc lies neither determines nor depends on the focus space in which the node or arc lies. To clarify the differences between these two spaces, we need to consider how focus spaces are established.

The procedural net representation of a task encodes both the subtask hierarchy and the partial ordering of subtask performance for that task. For any given execution of the task, only a subset of the nodes in the procedural net is invoked. These correspond to the subtasks actually discussed by the apprentice and the expert. For example, if the expert directs the apprentice to attach the belt housing cover, and the apprentice replies by saying that he has done it, then the nodes that correspond to details of how to perform the attaching are never invoked.

A new focus space is created for each subtask that enters the dialog. The procedural net imposes a hierarchical ordering on these spaces. This hierarchy will be used, as the logical one is, to determine what nodes and arcs are visible from a given space. Note, in particular, that the arcs and nodes that belong to a space are the only ones immediately visible from that space. Arcs and nodes in spaces that are above a given space also are potentially visible, but must be requested specifically to be seen. Other arcs and nodes are not visible.

The focus partitioning differs from the logical partitioning in several ways. First, a node may appear in any number of focus spaces but must appear in exactly one logical space. When the same object is used in two different subtasks (e.g., the wrench of Figure VI-12), either the same or different aspects of the object may be in focus in the two subtasks. It is also possible for a node or arc to be in no focus space. In this case, the object is not strongly associated with the performance of any particular subtask. For completeness, we define a top-most space, called the 'communal space', and a bottom-most space, called the 'vista space'. The communal space contains the relationships that are time-invariant (e.g., the fact that tools are found in tool boxes) or common to all contexts. The vista space is below all other spaces and hence can see everything in the semantic net. This perspective is useful for determining all the relationships into which an object has entered.

Figure VI-13 shows the net of Figure VI-12 with a focus partitioning superimposed on the logical partitioning. Focus Fi views wrench W as a box-end wrench that is being used in the operation of bolting the pump to the platform. Focus Fj views the same wrench as one that is in the apprentice's left hand. The other information about the wrench (e.g., its size) is recorded in the communal space. All the information is visible from the vista space.

The representation of an object in a focus space will include only the relationships that have been mentioned in the dialog concerning the corresponding subtask or that are inherent in the procedural net description of the local task. The distinction between the verbal context and the general world knowledge context, mentioned previously, may now be seen. The verbal context is supplied by the information recorded in the subspace hierarchy. The general world knowledge context is information that is present in the communal space. When resolving a reference, we can decide how to divide effort between examining links in the local space and looking back into the communal space.

Another advantage of adding this new partitioning is that special information can be recorded at the local focus level.

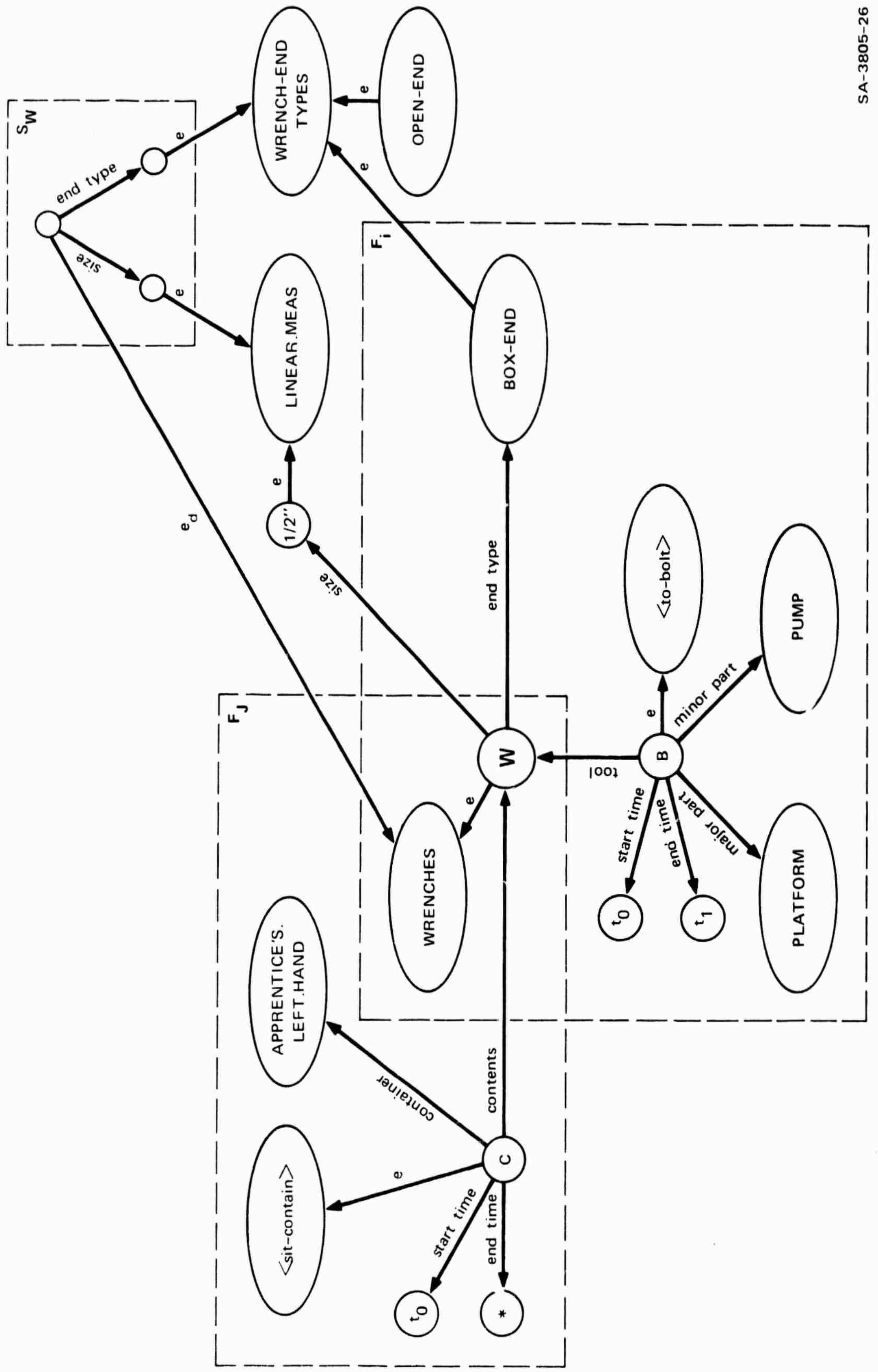


FIGURE VI-13 SEMANTIC NET FOR WRENCH, W, WITH FOCUS SPACE AND LOGIC PARTITIONINGS

Thus, if several links in the net must be followed to establish some fact about an object (i.e., some logical deduction must be done), the result of that work may be stored explicitly in the local focus space. The logical deduction does not have to be redone for local references. If this information is put in its own logic space, then it remains invisible from the knowledge net (the topmost logic space). For example, consider the situation portrayed in Figure VI-14. All the nodes and arcs in this figure are in one focus space. B-E is a set of box-end wrenches to which W1 belongs. H-E is a set of hex-end wrenches to which W2 belongs. If the apprentice now says, "... the box-end wrench", he means W1. The utterance level structure (created by parsing) for the phrase "the box-end wrench" is shown in Figure VI-15, and some amount of work must be done to establish the correspondence between W1 and W3. However, it is quite likely that W1 will again be referred to as "the box-end wrench". By explicitly storing the box-end property of W1 in the focus space, redundant work may be avoided. Figure VI-16 illustrates the new structure. Note that the e arc to WRENCHES and the end-type arc to BOX-END are in a separate logic space, L1. This makes them invisible at the knowledge net level. In fact, they are not visible from any logic partition outside this focus space.

Our experience with focus space partitioning and with other possible uses of network partitioning is limited. However, it is clear that the concept will prove to be extremely valuable in further work on discourse analysis and pragmatics.

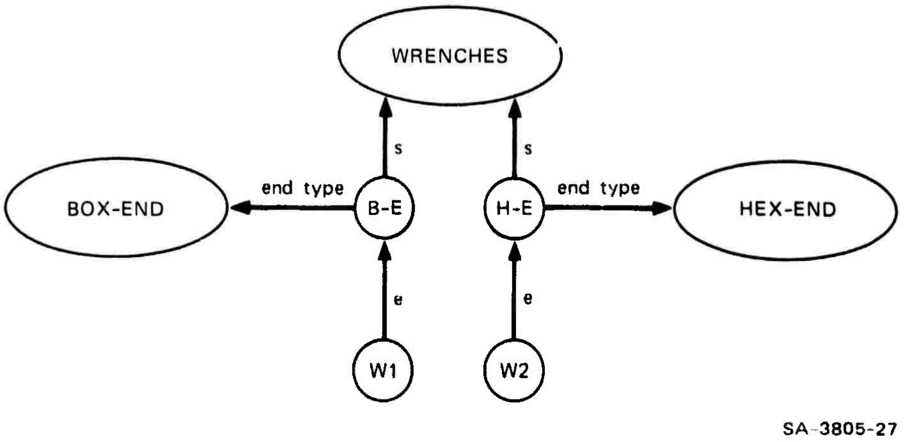


FIGURE VI-14 SEMANTIC NET SHOWING MEMBERS OF TWO SUBSETS OF THE SET "WRENCHES"

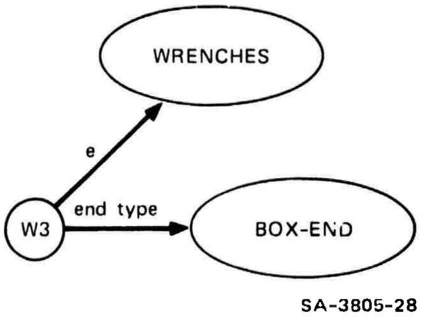


FIGURE VI-15 SEMANTIC NET FROM PARSE FOR "BOX-END WRENCH"

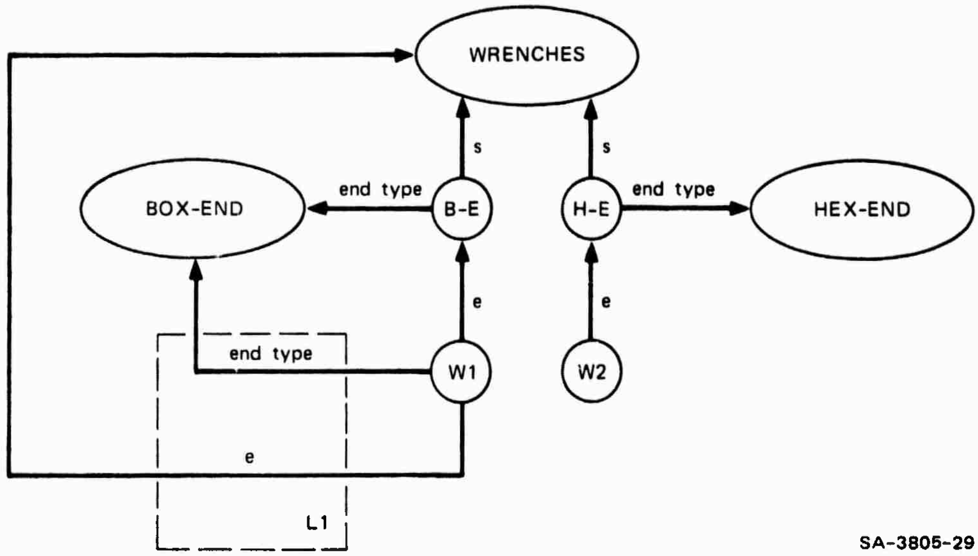


FIGURE VI-16 SEMANTIC NET SHOWING LOCAL FOCUS INFORMATION FOR BOX-END WRENCH, W1

VII REFERENCES

Backus, John W. The Syntax and Semantics of the Proposed International Algebraic Language of the Zurich ACM-GAMM Conference. Proceedings of the International Conference on Information Processing, UNESCO, 1959, 125-132.

Barnett, Jeffrey A., and Pintar, Douglas L. CRISP: A Programming Language and System. System Development Corporation, Santa Monica, California, 1974.

Becker, Richard, and Poza, Fausto. Acoustic Processing in the SRI Speech Understanding System. IEEE Transactions on Acoustics, Speech and Signal Processing, 1975 (in press).

Bever, Thomas G. On the Cognitive Basis for Linguistic Structures. In: Cognition and the Development of Language. Edited by John R. Hayes. Wiley, New York, 1970, 279-362.

Bresnan, Joan. Syntax of the Comparative Clause Construction in English. Linguistic Inquiry, 1973, 4, 275-343.

Cedergren, Henrietta J., and Sankoff, David. Variable Rules: Performance as a Statistical Reflection of Competence. Language, 1974, 50, 333-335.

Chafe, Wallace L. Discourse Structure and Human Knowledge. In: Language Comprehension and the Acquisition of Knowledge. Edited by Roy O. Freedle and John B. Carroll. V. H. Winston, Washington, D. C., 1972.

Chapanis, Alphonse. Interactive Human Communication. Scientific American, March 1975, 36-42.

Chomsky, Noam. Deep Structure, Surface Structure, and Semantic Interpretation. In: Semantics. Edited by Danny D. Steinberg and Leon A. Jakobovits. Cambridge University Press, Cambridge, 1971, 183-216.

Codd, Edward F. A Relational Model of Data for Large Shared Data Banks. Communications of the ACM, 1970, 13, 377-387.

Deutsch, Barbara G. The Structure of Task-Oriented Dialogs. Contributed Papers, IEEE Symposium on Speech Recognition, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 15-19 April 1974. IEEE, New York, 1974, 250-254.

Enea, Horace and Colby, Kenneth M. Idiolectic Language-Analysis for Understanding Doctor-Patient Dialogs. Advance Papers, Third International Joint Conference on Artificial Intelligence, Stanford, California, 20-23 August 1973. Stanford Research Institute, Menlo Park, California, 1973, 278-284.

Feldman, Jerome, and Gries, David. Translator Writing Systems. Communications of the ACM, 1968, 11, 77-113.

Fillmore, Charles J. The Case for Case. In: Universals in Linguistic Theory. Edited by Emmon Bach and Robert T. Harms. Holt, Rinehart and Winston, New York, 1968, 1-88.

Fromkin, Victoria A. The Non-Anomalous Nature of Anomalous Utterances. Language, 1971, 47, 27-52.

Grishman, Ralph. Implementation of the String Parser of English. In: Natural Language Processing. Edited by Randall Rustin. Algorithmics Press, New York, 1973, 89-109.

Grosu, Alexander. The Strategic Content of Island Constraints. Working Papers in Linguistics, No. 13. Department of Linguistics, Ohio State University, Columbus, Ohio, 1972, 1-225.

Hart, Peter E. Progress on a Computer Based Consultant. Technical Note 99, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California, January 1975.

Hendrix, Gary G. Expanding the Utility of Semantic Networks Through Partitioning. Advance Papers, International Joint Conference on Artificial Intelligence, Tbilisi, Georgian SSR, 3-8 September 1975.

Hobbs, Jerry R. A Metalanguage for Expressing Grammatical Restrictions in Nodal Spans Parsing of Natural Language. Courant Institute of Mathematical Sciences, New York, 1974.

Kaplan, Ronald M. A Multi-Processing Approach to Natural Language. Proceedings, National Computer Conference, New York, New York, 4-8 June 1973. Volume 42. AFIPS Press, New Jersey, 1973, 435-440.

Kimball, John. Seven Principles of Surface Structure Parsing in Natural Language. Cognition, 1973, 2, 15-47.

Lakoff, George. On Generative Semantics. In: Semantics. Edited by Danny D. Steinberg and Leon A. Jakobovits. Cambridge University Press, Cambridge, 1971, 232-296.

Lakoff, George. Fuzzy Grammar and the Performance/ Competence Terminology Game. In: Papers from the Ninth Regional Meeting. Edited by Claudia Corum, T. Cedric Smith-Stark, and Ann Wiser. Chicago Linguistic Society, 1973, 271-291.

Lesser, Victor R., Fennell, Richard D., Erman, Lee D., and Reddy, D. Raj. Organization of the Hearsay II Speech Understanding System. Contributed Papers, IEEE Symposium on Speech Recognition, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 15-19 April 1974. IEEE, New York, 1974, 11-21.

Miller, Perry L. A Locally Organized Parser for Spoken Input. Communications of the ACM, 1974, 17, 621-630.

Newell, Allen, et al. Speech Understanding Systems. North-Holland Publishing Company, Amsterdam, 1973.

Nilsson, Nils J. Problem Solving Methods in Artificial Intelligence. McGraw-Hill, New York, 1971.

Nilsson, Nils J., et al. Artificial Intelligence--Research and Applications. Annual Report, Project 1530, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California, April 1974.

Nilsson, Nils J., et al. Artificial Intelligence--Research and Applications. Annual Report, Project 3805, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California, May 1975.

Paxton, William H. A Best-First Parser. Contributed Papers, IEEE Symposium on Speech Recognition, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 15-19 April 1974. IEEE, New York, 1974, 218-225. [IEEE Transactions on Acoustics, Speech and Signal Processing, in press.]

Paxton, William H., and Robinson, Ann E. A Parser for a Speech Understanding System. Advance Papers, International Joint Conference on Artificial Intelligence, Stanford, California, 20-23 August 1973. Stanford Research Institute, Menlo Park, California, 1973, 216-222.

Riesbeck, Christopher Kevin. Computational Understanding. Memo AIM-238, Stanford Artificial Intelligence Laboratory, Stanford University, Stanford, California, 1974.

Ritea, H. Barry. A Voice-Controlled Data Management System. Contributed Papers, IEEE Symposium on Speech Recognition, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 15-19 April 1974. IEEE, New York, 1974, 28-31.

Robinson, Jane J. Performance Grammars. Invited Papers, IEEE Symposium on Speech Recognition, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 15-19 April 1974. To be published by Academic Press, New York, 1975.

Ross, John Robert. Constraints on Variables in Syntax. Ph. D. Dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1967. [Available from Indiana University Linguistics Club.]

Ross, John Robert. Gapping and the Order of Constituents. In: Progress in Linguistics, Edited by Manfred Bierwisch and Karl Erich Heidolph. Mouton, The Hague, 1970, 249-259.

Ross, John Robert. The Category Squish: Endstation Hauptwort. Papers from the Eighth Regional Meeting. Edited by Paul M. Pertanteau, Judith N. Levi, and Gloria C. Phares. Chicago Linguistic Society, 1972, 316-328.

Ross, John Robert. A Fake NP Squish. In: New Ways of Analyzing Variation in English. Edited by Charles-James N. Bailey and Roger Shuy. Washington, D.C., Georgetown University Press, 1973, 96-140. (a)

Ross, John Robert. Nouniness. In: Three Dimensions of Linguistic Theory. Edited by Osamu Fujimura. TEC, Tokyo, 1973. (b)

Rumelhart, David E., and Norman, Donald A. Active Semantic Networks as a Model of Human Memory. Advance Papers, International Joint Conference on Artificial Intelligence, Stanford, California, 20-23 August 1973. Stanford Research Institute, Menlo Park, California, 1973, 450-457.

Rustin, Randall (Ed.). Natural Language Processing. Algorithmics Press, New York, 1973.

Sacerdoti, Earl D. A Structure for Plans and Behavior. Ph. D. Thesis, Stanford University, Stanford, California, forthcoming.

Sager, Naomi. The String Parser for Scientific Literature. In: Natural Language Processing. Edited by Randall Rustin. Algorithmics Press, New York, 1973. Pp. 27-52.

Schank, Roger C. Identification of Conceptualizations Underlying Natural Language. In: Computer Models of Thought and Language. Edited by Roger C. Schank and Kenneth M. Colby. W. H. Freeman, San Francisco, 1973, 187-247.

Schank, Roger C., and Tesler, Lawrence G. A Conceptual Parser for Natural Language. Proceedings of the International Joint Conference on Artificial Intelligence, Washington, D.C., 7-9 May 1969. Edited by Donald E. Walker and Lewis M. Norton. The MITRE Corporation, Bedford, Massachusetts, 1969, 569-578.

Shapiro, Stuart C. A Net Structure for Semantic Information Storage, Deduction and Retrieval. Advance Papers, Second International Joint Conference on Artificial Intelligence, London, England, 1-3 September 1971. The British Computer Society, London, England, 1971, 512-523.

Simmons, Robert F. Semantic Networks: Their Computation and Use for Understanding English Sentences. In: Computer Models of Thought and Language. Edited by Roger C. Schank and Kenneth M. Colby. W. H. Freeman, San Francisco, 1973, 63-113.

Teitelman, Warren. INTERLISP Reference Manual. XEROX Palo Alto Research Center, Palo Alto, California, 1974.

Thorne, James P., Bratley, Paul, and Dewar, Hamish. The Syntactic Analysis of English by Machine. In: Machine Intelligence 3. Edited by Donald Michie. Edinburgh University Press, Edinburgh, and American Elsevier, New York, 1968.

Walker, Donald E. Speech Understanding Research. Annual Report, Project 1526, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California, February 1973. (a)

Walker, Donald E. Speech Understanding Through Syntactic and Semantic Analysis. Advance Papers, International Joint Conference on Artificial Intelligence, Stanford, California, 20-23 August 1973. Stanford Research Institute, Menlo Park, California, 1973, 208-215. (b)

Walker, Donald E. Speech Understanding, Computational Linguistics, and Artificial Intelligence. In: Computational and Mathematical Linguistics, Proceedings of the International

Conference on Computational Linguistics, Volume I, Edited by Antonio Zampolli. Casa Editrice Leo S, Olschki, Firenze, 1973, (c)

Walker, Donald E. Speech Understanding Research. Annual Report, Project 1526, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California, May 1974. (a)

Walker, Donald E. The SRI Speech Understanding System. Contributed Papers, IEEE Symposium on Speech Recognition, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 15-19 April 1974. IEEE, New York, 1974, 32-37. [IEEE Transactions on Acoustics, Speech and Signal Processing, in press.] (b)

Winograd, Terry. Procedures as a Representation for Data in a Computer Program for Understanding Natural Language. Report MAC-TR-84, Project MAC, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1971. [Published as Understanding Natural Language, Academic Press, New York, 1972.]

Winograd, Terry. Frame Representations and the Declarative/Procedural Controversy. In: Representation and Understanding. Edited by Daniel G. Bobrow and Allen M. Collins. Academic Press, New York, 1975.

Woods, William A. Transition Network Grammars for Natural Language Analysis. Communications of the ACM, 1970, 13, 591-606.

Woods, William A. An Experimental Parsing System for

Transition Network Grammars, In: Natural Language Processing, Edited by Randall Rustin, Algorithmics Press, New York, 1973, 1[illegible]154.

Woods, William A. Motivation and Overview of BBN SPEECHLIS, An Experimental Prototype for Speech Understanding Research. Contributed Papers, IEEE Symposium on Speech Recognition, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 15-19 April 1974. IEEE, New York, 1974, 1-10.

APPENDIX A LANGUAGE DEFINITION

Prepared by Jane J. Robinson and Ann E. Robinson

Contents:


```
    INFIX FILE LANGDF.GRM

    SECTION(71,"(71 0));


LANGUAGE.DEFINITION
    CATEGORIES U,N,NOUN,NP,DET,ART,BE,DO,VERB,V,QUANT,PREP,REL,MP,
        THANR,DIGIT,NUMBER,TEEN,TOKEN,S,NOM,AUXD,AUXB,NEG,NUMBERP,
        VP,ADJ,NOMHEAD,PREPP,SMALLNUM,DIGTY,BIGADD,BIGMULT,BIGCAT;
    ROOT CATEGORY U;
    AFFIXES PL,SG,TEEN,TY,NT,PPL,GEN;
    RULEFN RULEFN;
    WORDFN WORDFN;
    CATEGORYFN CATEGORYFN;
    RESPONSEFN RESPONSE;

ATTRIBUTES  %'DECLARE ATTRIBUTES FOR THE VARIOUS CATEGORIES'
    ALL HAVE MAPINFO, PHRMAPINFO;
    ALL HAVE LEFT,RIGHT,STRING,FSTWD,LSTWD,SPELLING;
    ALL HAVE SIZE,DEPTH,BULK;
```

```
    ALL EXCEPT TOKEN HAVE SEMANTICS;
    NP,MP,NUMBERP HAVE WDSEMANTICS;
    S,VP, VERB HAVE VOICE;
    U,NP,PREPP,DO,QUANT,S,AUXB,AUXD HAVE AFFNEG;
    DIGIT HAS DIGTYP;
    U,NP HAVE ELLIPSE;
    U,VP,NOMHEAD,NUMBER,MP,S,NP,NUMBERP,DET,ART,NUMBER,PREPP HAVE
        MOOD;
    S,VP,NP,VERB,V HAVE TRANS;
    NP,VP,AUXD,AUXB,DO,BE,NOM,NUMBERP,NUMBEP, DET,ART,
    NOMHEAD,QUANT,N,NOUN,MP,SMALLNUM,DIGIT,TEEN,PREPP,DIGTY,VERB
        HAVE NBR;
    NP,DET HAVE GCASE;
    AUXD,AUXB,DO,BE,NEG HAVE STRESS;
    U,NUMBERP,MP,QUANT,S,VP,PREPP,NP,DET,ART,NUMBER HAVE FOCUS;
    VP,VERB,V HAVE IMP;
    S,U HAS PITCHC;
    NP,AUXB,BE HAVE PERS;
    VP,VERB,V HAVE AGENCY;
    ADJ,DET,NP,NOM,NOMHEAD,NOUN,N HAVE SUBCAT;
    NP,QUANT,NOM,NOMHEAD,NOUN,NUMBERP,NUMBER,PREPP,S,
        DET,ART,N,MP HAVE CMU;
    NP,BIGCAT,BIGADD,BIGMULT,MP,NUMBER,SMALLNUM,DIGIT,DIGTY,TEEN,
        NUMBERP HAVE NUM;
    N HAS PLSUFF;
    ADJ HAS MARK;
    ADJ HAS CFORM;
    N,NOUN,NOMHEAD,NOM,NP,S,PREPP HAVE RELN;
    NP HAS GENSUFF;
    ADJ HAS CVAL;
    THANR HAS REL;
    SMALLNUM,TEEN HAVE NUMTYP;
    PREPP,PREP HAVE SEMPREP;

    ENDATTRS;
END;


EOF
```

```
    INFIX FILE TOKEN.LEX

    SECTION (71,"(71 0));


CATEGORY.DEF    TOKEN

 END;


WORDS.DEF    TOKEN
    AND;

    OF;

    HOW;

    NOT;

    THAN;

    %'SUFFIXES DO NOT APPEAR IN THE LEXICON; ONLY IN THE RULES.
    AT PRESENT, SUFFIXES ARE: -GEN, -NT, -PL, -SG, -TEEN,
    AND -TY.'

 ENDWORDS;


EOF
```

```
    INFIX FILE NOUN.LEX

    SECTION(71,"(71 0));

        %'LEXICON'

    %'NOUNS'

CATEGORY.DEF    N

    FACTORS
        INIT = 80,
        RESCHEDULE;

    WORDFN
        LAMBDA (CAT, WORDDEF)
        BEGIN
            IF NOT (ATTRCK("CMU)) THEN ADDATTR("CMU, "(COUNT));
            RETURN WORDDEF;
        END;
        %'SETS CMU = (COUNT) UNLESS OTHERWISE DEFINED'

 END;

WORDS.DEF    N

    DIESEL
        SEMANTICS = ((SUPSET #DIESELS)(CMU COUNT)(NBR S));

    DRAFT
        RELN = T,
        SEMANTICS = ((SUPSET #DRAFTS)(CMU COUNT)(NBR S)
            (INVERSIONS(((SUPSET #DRAFT.RELS)
            (CASES((MEASURE *)))(PDGM PG.BINATT)))));

    ETHAN.ALLEN
        SEMANTICS = ((SUPSET #ETHAN.ALLENS)(CMU COUNT)
            (NBR S));

    FOOT
        PLSUFF = NO,
        CMU = (UNIT),
        SEMANTICS = ((SUPSET #FOOT)(MEASURES #LINEAR.MEAS));

    FUEL
        CMU = (MASS);

    GUPPY.THREE
        SEMANTICS = ((SUPSET #GUPPY3S)(CMU COUNT)(NBR S));
```

```
KNOT
    CMU = (UNIT),
    SEMANTICS = ((SUPSET #KNOT)(CMU UNIT)(NBR S)
        (MEASURES #SPEED,MEAS));

LAFAYETTE
    SEMANTICS = ((SUPSET #LAFAYETTES)(CMU COUNT)
        (NBR S));

LENGTH
    RELN = T,
    SEMANTICS = ((SUPSET #LENGTHS)(CMU COUNT)(NBR S)
        (INVERSIONS(((SUPSET #LENGTH,RELS)
        (CASES((MEASURE *)))(PDGM PG,BINATT)))));

NUC
    SEMANTICS = ((SUPSET #NUCS)(CMU COUNT)
        (NBR S));

SEAWOLF
    SEMANTICS = ((SUPSET #SEAWOLFS)(CMU COUNT)(NBR S));

SPEED
    RELN = T,
    SEMANTICS = ((SUPSET #SPEEDS)(CMU COUNT)(NBR S)
        (INVERSIONS(((SUPSET
        #SPEED,RELS)(CASES((MEASURE *)))(PDGM PG,BINATT)))));

SUBMARINE
    SEMANTICS = ((SUPSET #SUBS)(CMU COUNT)(NBR S));

SUBMERGED,DISPLACEMENT
    RELN = T,
    SEMANTICS = ((SUPSET #SUBM,DISPS)(CMU COUNT)
        (NBR S)(INVERSIONS(((SUPSET #SUBM,DISP,RELS)
        (CASES((MEASURE *)))(PDGM PG,BINATT)))));

SURFACE,DISPLACEMENT
    RELN = T,
    SEMANTICS = ((SUPSET #SURF,DISPS)(CMU COUNT)
        (NBR S)(INVERSIONS(((SUPSET #SURF,DISP,RELS)
        (CASES((MEASURE *)))(PDGM PG,BINATT)))));

SUBMERGED,SPEED
    RELN = T,
    SEMANTICS = ((SUPSET #SUBM,SPEEDS)(CMU COUNT)
        (NBR S)(INVERSIONS((
        (SUPSET #SUBM,SPEED,RELS)(CASES((MEASURE *)))
        (PDGM PG,BINATT)))));

SURFACE,SPEED
    RELN = T,
    SEMANTICS = ((SUPSET #SURF,SPEEDS)(CMU COUNT)
```

```
            (NBR S)(INVERSIONS((
            (SUPSET #SURF.SPEED.RELS)(CASES((MEASURE #)))
            (PDGM PG.BINATT)))));

    TON
        CMU = (UNIT),
        SEMANTICS = ((SUPSET #TON)(MEASURE #MEASURE.DISP));

    TORPEDO.TUBE
        SEMANTICS = ((SUPSET #TORPEDO.TUBES)(CMU COUNT)(NBR S));

    U.S.
        PLSUFF = NO,
        SUBCAT = PROPN,
        SEMANTICS = ((SUPSET #USAS)(CMU COUNT)(NBR S));

 ENDWORDS;


CATEGORY.DEF    NOUN

 END;


WORDS.DEF    NOUN

    FEET
        CMU = (UNIT),
        NBR = (PL),
        SEMANTICS = ((SUPSET #FOOT)(CMU UNIT)(NBR PL)
            (MEASURES #LINEAR.MEAS));
 ENDWORDS;


CATEGORY.DEF    NP

    ATTRIBUTES
        PLSUFF = "NO,
        SUBCAT = "PRO,
        SEMANTICS = SEMCALL("SEMRNP5,WDSEMANTICS,NBR);

    FACTORS
        INIT = 80,
        RESCHEDULE;

 END;


WORDS.DEF    NP


    I
        MOOD = (DEC),
```

```
      FOCUS = (DEF),
      GCASE = (NOM),
      CMU = (COUNT),
      NBR = (SG),
      PERS = EGO;

 IT
      MOOD = (DEC),
      FOCUS = (DEF INDEF),
      GCASE = (NOM ACC),
      CMU = (COUNT MASS UNIT),
      NBR = (SG),
      PERS = 3,
      WDSEMANTICS = (AMBIGUOUS ((SUPSET #UNIOBJS)(NBR S)
           (ISF ISF))
           ((SUPSET #UNIOBJS,MASS)(NBR M)(ISF ISF)));

 ME
      MOOD = (DEC),
      FOCUS = (DEF),
      GCASE = (ACC),
      CMU = (COUNT),
      NBR = (SG),
      PERS = EGO;

 THEY
      MOOD = (DEC),
      FOCUS = (DEF INDEF),
      GCASE = (NOM),
      CMU = (COUNT MASS UNIT),
      NBR = (PL),
      PERS = 3,
      WDSEMANTICS = ((SUPSET #UNIOBJS)(NBR S)(ISF ISF));

 THEM
      MOOD = (DEC),
      FOCUS = (DEF INDEF),
      GCASE = (ACC),
      CMU = (COUNT MASS UNIT),
      NBR = (PL),
      WDSEMANTICS = ((SUPSET #UNIOBJS)(NBR S)(ISF ISF));


 US
      MOOD = (DEC),
      FOCUS = (DEF),
      GCASE = (ACC),
      CMU = (COUNT),
      NBR = (PL),
      WDSEMANTICS = ((SUPSET #USAS)(CMU COUNT)(NBR S));

 WE
      MOOD = (DEC),
```

```
        FOCUS = (DEF),
        GCASE = (NOM),
        CMU = (COUNT),
        NBR = (PL),
        PERS = EGO,
        WDSEMANTICS = ((SUPSET #USAS)(CMU COUNT)(NBR S));

    WHO
        NBR = (SG PL),
        MOOD = (WH REL),
        GCASE = (NOM),
        WDSEMANTICS = (AMBIGUOUS((E.QST)(SUPSET #LEGAL.PERSONS)
            (NBR PL)(DET ?)(ISF ISF))
            ((E.QST)(SUPSET #LEGAL.PERSONS)
            (NBR S)(DET ?)(ISF ISF)));

    WHOM
        NBR = (SG PL),
        GCASE = (NOM),
        WDSEMANTICS = (AMBIGUOUS ((E.QST)(SUBSET #LEGAL.PERSONS)
            (NBR PL)(DET ?)(ISF ISF))
            ((E.QST)(SUPSET #LEGAL.PERSONS)
            (NBR S)(DET ?)(ISF ISF)));

 ENDWORDS;

EOF
```

```
    INFIX FILE DETERM.LEX

    SECTION(71, "(71 0));


    %'DETERMINERS AND ARTICLES'


CATEGORY.DEF    DET

    ATTRIBUTES
        FOCUS = "DEF;

    FACTORS
        INIT = 80,
        WD = IF SPELLING EQ "WHAT THEN
            IF LEFT EQUAL STARTTIMEBOUNDARY THEN GOOD ELSE POOR
            ELSE OK,
        RESCHEDULE;

 END;


WORDS.DEF    DET

    ITS
        CMU = (COUNT MASS UNIT),
        MOOD = (DEC),
        SUBCAT = PRO,
        GCASE = (GEN),
        SEMANTICS = (AMBIGUOUS((SUPSET #UNIOBJS)(NBR S)(ISF ISF))
            ((SUPSET #UNIOBJS.MASS)(NBR M)(ISF ISF)));

    THAT
        NBR = (SG),
        MOOD = (DEC),
        CMU = (COUNT MASS UNIT),
        SEMANTICS =  ((SUBTYPE (SET 1 2))(DET DEF)(NBR(SET M S)));

    THESE
        NBR = (PL),
        MOOD = (DEC),
        CMU = (COUNT UNIT),
        SEMANTICS = ((SUBTYPE (SET 1 2))(DET DEF)(NBR PL));

    THIS
        MOOD = (DEC),
        NBR = (SG),
        CMU = (COUNT MASS UNIT),
        SEMANTICS = ((SUBTYPE (SET 1 2))(DET DEF)(NBR (SET M S)));

    THOSE
        NBR = (PL),
```

```
        CMU = (COUNT UNIT),
        SEMANTICS = ((SUBTYPE (SET 1 2))(DET DEF)(NBR PL));

    WHAT
        MOOD = (WH),
        CMU = (COUNT MASS UNIT),
        NBR = (SG PL),
        SEMANTICS = ((E.QST)(TYPE WH)(SUBTYPE (SET 1 2))(DET ?)
            (NBR (SET M PL S)));

    WHICH
        MOOD = (WH REL),
        CMU = (COUNT UNIT),
        NBR = (SG PL),
        SEMANTICS = ((E.QST)(TYPE WH)(SUBTYPE(SET 1 2 3))(DET ?)
            (NBR (SET M PL S)));

    WHOSE
        CMU = (COUNT MASS UNIT),
        SUBCAT = PRO,
        GCASE = (GEN),
        MOOD = (WH),
        SEMANTICS = (AMBIGUOUS((E.QST)(SUPSET #LEGAL.PERSONS)
            (NBR PL)(DET ?)(ISF ISF))
            ((E.QST)(SUPSET #LEGAL.PERSONS)(NBR S)
            (DET ?)(ISF ISF)));

 ENDWORDS;


CATEGORY.DEF    ART

 END;


WORDS.DEF   ART

    A
        MOOD = (DEC),
        CMU = (COUNT UNIT),
        NBR = (SG),
        SEMANTICS = ((DET INDEF)(NBR (SET M S)));

    THE
        NBR = (PL SG),
        CMU = (COUNT MASS UNIT),
        MOOD = (DEC);

 ENDWORDS;


EOF
```

```
    INFIX FILE NUMBERS.LEX

    SECTION (71,"(71 0));


    $'NUMBERS'


CATEGORY.DEF    DIGIT

 END;


WORDS.DEF    DIGIT

    ONE
        DIGTYP = (1),
        NUM = 1;

    TWO
        DIGTYP = (1),
        NUM = 2;

    THREE
        DIGTYP = (1),
        NUM = 3;

    FOUR
        DIGTYP = (1 2 3),
        NUM = 4;

    FIVE
        DIGTYP = (1),
        NUM = 5;

    SIX
        DIGTYP = (1 2 3),
        NUM = 6;

    SEVEN
        DIGTYP = (1 2 3),
        NUM = 7;

    EIGHT
        DIGTYP = (1 2 3),
        NUM = 8;

    NINE
        DIGTYP = (1 2 3),
        NUM = 9;

    TWEN
        DIGTYP = (3),
```

```
        NUM = 2;

    THIR
        DIGTYP = (2 3),
        NUM = 3;

    FIF
        DIGTYP = (2 3),
        NUM = 5;

 ENDWORDS;


CATEGORY.DEF    BIGCAT

 END;


WORDS.DEF    BIGCAT

    HUNDRED
        NUM = 100;

    THOUSAND
        NUM = 1000;

    MILLION
        NUM = 1000000;

    BILLION
        NUM = 1000000000;

 ENDWORDS;


CATEGORY.DEF    TEEN

 END;


WORDS.DEF    TEEN

    TEN
        NUMTYP = DECADE,
        NUM = 10;

    ELEVEN
        NUMTYP = DECADEPLUS,
        NUM = 11;

    TWELVE
        NUMTYP = DECADEPLUS,
        NUM = 12;
```

```
ENDWORDS;

EOF
```

```
    INFIX FILE THANR.LEX

    SECTION (71,"(71 0));


    %'THANR WORDS'

CATEGORY.DEF    THANR

 END;

WORDS.DEF    THANR

    FEWER
        REL = LESSTHAN;

    GREATER
        REL = GREATERTHAN;

    LESS
        REL = LESSTHAN;

    MORE
        REL = GREATERTHAN;

 ENDWORDS;

EOF
```

```
    INFIX FILE VERBS.LEX

    SECTION (71,"(71 0));


    %'VERBS'


CATEGORY.DEF    BE

 END;


WORDS.DEF    BE

    AM
        NBR = (SG),
        PERS = EGO;

    ARE
        NBR = (PL),
        SEMANTICS = ((NBR PL));

    IS
        NBR = (SG),
        PERS = 3,
        SEMANTICS = ((NBR (SET M S)));

 ENDWORDS;


CATEGORY.DEF    DO

 END;


WORDS.DEF    DO

    DO
        NBR = (PL),
        SEMANTICS = ((NBR PL));

    DOES
        NBR = (SG),
        SEMANTICS = ((NBR S));

    DONT
        NBR = (SG PL),
        AFFNEG = NEG,
        SEMANTICS = ((NBR (SET S PL)));

 ENDWORDS;
```

```
CATEGORY.DEF    VERB

    FACTORS
        INIT = 80,
        RESCHEDULE;

 END;


WORDS.DEF    VERB

    HAS
        NBR = (SG),
        TRANS = 2,
        AGENCY = NO,
        IMP = NO,
        SEMANTICS = (AMBIGUOUS ((SUPSET #OWN.RELS)(NBR S)
            (SEMROOT OWN)(PDGM PG.TRANS)(MANDATORY OWN;ACTOR
            OWN;DO))((SUPSET #HAS.PART.RELS)(PDGM PG.TRANS)
            (SEMROOT HAVEPART)
            (MANDATORY (HAVEPART;ACTOR HAVEPART;DO)))
            ((SUPSET #HAS.PART.RELS)(PDGM (HAAFN))));

    HAVE
        NBR = (PL),
        TRANS = 2,
        AGENCY = NO,
        IMP = NO,
        SEMANTICS = (AMBIGUOUS ((SUPSET #OWN.RELS)(NBR S)
            (SEMROOT OWN)(PDGM PG.TRANS)(MANDATORY OWN;ACTOR
            OWN;DO))((SUPSET #HAS.PART.RELS)(PDGM PG.TRANS)
            (SEMROOT HAVEPART)
            (MANDATORY (HAVEPART;ACTOR HAVEPART;DO)))
            ((SUPSET #HAS.PART.RELS)(PDGM (HAAFN))));

 ENDWORDS;


CATEGORY.DEF    V

 END;


WORDS.DEF    V

    LIST
        TRANS = 2,
        AGENCY = NO,
        IMP = YES;

    OWN
        TRANS = 2,
        AGENCY = YES,
```

```
        IMP = NO,
        SEMANTICS = ((SUPSET #OWN.RELS)(PDGM PG.TRANS)
            (MANDATORY (OWN;ACTOR OWN;DO)));

 ENDWORDS;

EOF
```

```
    INFIX FILE PREP.LEX

    SECTION (71,"(71 0));

    %'PREPS'

CATEGORY.DEF    PREP

 END;

WORDS.DEF    PREP

    BY
        SEMPREP = PG.BY;

    OF
        SEMPREP = PG.OF;

    WITH
        SEMPREP = PG.WITH;

 ENDWORDS;

EOF
```

```
    INFIX FILE QUANT.LEX

    SECTION (71,"(71 0));


    %'QUANTIFIERS'


CATEGORY.DEF    QUANT

 END;


WORDS.DEF    QUANT

    ALL
        CMU = (COUNT MASS UNIT),
        NBR = (SG PL),
        SEMANTICS = ((SUBETYP(SET 1 2 3 4))(QUANTF FOREVERY)
            (NBR (SET M PL))(NMOD((DET (SET DEF UNI))
            (NBR(SET M PL))))(ALL ALL)(NUMREST (LESSP 2)));

    ANY
        CMU = (COUNT MASS UNIT),
        NBR = (SG PL),
        SEMANTICS = ((SUBTYPE (SET 1 2 3 4))(QUANTF(SET FOREVERY
            CHOICE))(NBR(SET S PL M))(NMOD((DET(SET DEF UNI))(NBR
            (SET M PL))))(NUMREST(NUMBERP)));

    BOTH
        CMU = (COUNT UNIT),
        NBR = (PL),
        FOCUS = DEF,
        SEMANTICS = ((SUBTYPE(SET 1 2))(QUANTF FOREVERY)(NBR PL)
            (NMOD((DET DEF)(NBR PL)(NUM 2)))
            (NUMREST NOTAPPLICABLE));

    EACH
        CMU = (COUNT UNIT),
        NBR = (SG),
        SEMANTICS = ((SUBTYPE(SET 1 2 3))(QUANTF FOREVERY)(NBR S)
            (NMOD ((DET (SET DEF UNI))(NBR PL)))
            (NUMREST (EQ 1)));

    EITHER
        CMU = (COUNT),
        NBR = (SG),
        SEMANTICS = ((SUBTYPE(SET 1 2 3))(QUANTF CHOICE)
            (NBR S)(NMOD ((DET DEF)(NBR PL)(NUM 2)))
            (NUMREST(EQ 1)));

    EVERY
        CMU = (COUNT UNIT),
```

```
        NBR = (SG),
        SEMANTICS = ((SUBTYPE(SET 1 3))(QUANTF FOREVERY)(NBR S)
            (NMOD((DET(SET DEF UNI))(NBR PL)))(NUMREST (EQ 1)));

    NEITHER
        CMU = (COUNT),
        NBR = (SG),
        AFFNEG = NEG,
        SEMANTICS = ((SUBTYPE(SET 1 2 3))(QUANTF NOTANY)
            (NBR S)(NMOD((DET DEF)(NBR PL)(NUM 2)))
            (NUMREST (EQ 1)));

    NO
        CMU = (COUNT MASS UNIT),
        NBR = (SG PL),
        AFFNEG = NEG,
        SEMANTICS = ((SUBTYPE(SET 1 3 4))(QUANTF NOTANY)
            (NBR (SET M PL S))(NMOD((DET(SET DEF UNI))
            (NBR(SET M PL))))(NUMREST(NUMBERP)));

    NONE
        CMU = (COUNT UNIT),
        NBR = (SG PL),
        AFFNEG = NEG,
        SEMANTICS = ((SUBETYP 2)(QUANTF NOTANY)(NBR(SET M PL))
            (NMOD ((DET (SET DEF UNI))(NBR(SET M PL))))(NUMREST
            NOTAPPLICABLE));

    SOME
        CMU = (COUNT MASS UNIT),
        NBR = (SG PL),
        SEMANTICS = ((SUBTYPE(SET 1 2))(QUANTF CHOICE)(NBR
            (SET S M PL))(NMOD((DET (SET DEF UNI))
            (NBR (SET M PL)))));

 ENDWORDS;


EOF
```

```
INFIX FILE MPWORDS.LEX

SECTION (71,"(71 0));


%'MP'


CATEGORY.DEF    MP

    ATTRIBUTES
        FOCUS = "INDEF,
        SEMANTICS = SEMCALL("SEMRMP,WDSEMANTICS);

END;


WORDS.DEF    MP

    FEW
        NBR = (PL),
        NUM = FEW,
        WDSEMANTICS = ((NBR PL)(NET #NUMBER;FEW));

    LITTLE
        CMU = (MASS),
        NBR = (SG),
        NUM = LITTLE,
        WDSEMANTICS = ((NBR M)(NET #NUMBER;FEW));

    MANY
        NBR = (PL),
        NUM = MANY,
        WDSEMANTICS = ((NBR PL)(NET #NUMBER;MANY));

    MUCH
        CMU = (MASS),
        NBR = (SG),
        NUM = MUCH,
        WDSEMANTICS = ((NBR M)(NET #NUMBER;MANY));

ENDWORDS;


EOF
```

```
    INFIX FILE NMPWORDS.LEX

    SECTION (71,"(71 0));


    %'NUMBERP WORDS'


CATEGORY.DEF    NUMBERP

    ATTRIBUTES
        SEMANTICS = SEMCALL("SEMRMP,WDSEMANTICS);

    FACTORS
        INIT = 80,
        RESCHEDULE;

 END;


WORDS.DEF    NUMBERP

    HOW.MANY
        MOOD = (WH),
        FOCUS = INDEF,
        NBR = (PL),
        WDSEMANTICS = ((E.QST)(NUM ?)(NBR PL)
            (NET #NUMBER.QST));

 ENDWORDS;


CATEGORY.DEF    U

    FACTORS
        LEFT = COART(LEFT,STARTTIMEBOUNDARY),
        RIGHT = COART(RIGHT,ENDTIMEBOUNDARY);

 END;


WORDS.DEF    U

    OKAY
        AFFNEG = AFF,
        MOOD = (DEC);

    OK
        AFFNEG = AFF,
        MOOD = (DEC);

 ENDWORDS;
```

EOF

```
    INFIX FILE URULES.GRM

    SECTION(71, "(71 72 0 73));


    %'UTTERANCE LEVEL RULES'


RULE.DEF U1     U = S;

    ATTRIBUTES
        ELLIPSE = "NO,
        MOOD,FOCUS,AFFNEG FROM S,
        PHRMAPINFO = PHRM(STRING,
        STARTTIMEBOUNDARY,ENDTIMEBOUNDARY),
        SEMANTICS FROM S;

    FACTORS
        PROB =  LK1,
        LEFT = IF VIRTUAL THEN OK
            ELSE COART(LEFT,STARTTIMEBOUNDARY),
        RIGHT = IF VIRTUAL THEN OK
            ELSE COART(RIGHT,ENDTIMEBOUNDARY),
        MOOD = IF MOOD EQUAL "(WH) THEN VERYGOOD ELSE OK,
        SCORE IF NOT VIRTUAL,
        SIZE = [X=SIZE, IF X EQ "UNDEFINED THEN 60 ELSE
        60+(40*X)/DISTANCEBETWEENSTARTANDEND],
        PHRMAPPING = IF VIRTUAL THEN OK
        ELSE PMCHECK(PHRMAPINFO,STRING);

    EXAMPLES
        WHAT IS THE SURFACE DISPLACEMENT OF THE LAFAYETTE (OK);

 END;


RULE.DEF U2     U = NP;

    ATTRIBUTES
        ELLIPSE = "YES,
        FOCUS,MOOD FROM NP,
        PHRMAPINFO = PHRM(STRING,STARTTIMEBOUNDARY,
            ENDTIMEBOUNDARY),
        SEMANTICS FROM NP;

    FACTORS
        PROB = LK2,
        LEFT = IF VIRTUAL THEN OK
            ELSE COART(LEFT,STARTTIMEBOUNDARY),
        RIGHT = IF VIRTUAL THEN OK
            ELSE COART(RIGHT,ENDTIMEBOUNDARY),
        MOOD = [X=MOOD(NP),
            IF SUBCAT(NP) EQ "PRO AND X EQUAL "(DEC)
```

```
            THEN BAD ELSE IF X EQUAL "(WH) THEN VERYGOOD ELSE OK],
        SCORE IF NOT VIRTUAL,
        SIZE = [X=SIZE, IF X EQ "UNDEFINED THEN 60
            ELSE 60+(40*X)/DISTANCEBETWEENSTARTANDEND],
        PHRMAPPING = IF VIRTUAL THEN OK
            ELSE PMCHECK(PHRMAPINFO,STRING);

    EXAMPLES
        HOW MANY LAFAYETTES (OK)
        THE ETHAN ALLEN (OK)
        WHO (OK)
        WE (BAD);

 END;


RULE.DEF U3     U = NOM;

    ATTRIBUTES
        PHRMAPINFO = PHRM(STRING,STARTTIMEBOUNDARY,
            ENDTIMEBOUNDARY),
        SEMANTICS FROM NOM,
        ELLIPSE = "YES;

    FACTORS
        PROB = LK2,
        LEFT = IF VIRTUAL THEN OK
            ELSE COART(LEFT,STARTTIMEBOUNDARY),
        RIGHT = IF VIRTUAL THEN OK
            ELSE COART(RIGHT,ENDTIMEBOUNDARY),
        SCORE IF NOT VIRTUAL,
        SIZE = [X=SIZE, IF X EQ "UNDEFINED THEN 60 ELSE
            60+(40*X)/DISTANCEBETWEENSTARTANDEND],
        PHRMAPPING = IF VIRTUAL THEN OK
            ELSE PMCHECK(PHRMAPINFO,STRING);

    EXAMPLES
        SUBMERGED DISPLACEMENT (OK);

 END;


EOF
```

```
    INFIX FILE SRULES.GRM

    SECTION(71, "(71 72 0));


    %"SENTENCE LEVEL RULES'


RULE.DEF S1     S = NP VP;

    ATTRIBUTES
        FOCUS,MOOD FROM NP,
        VOICE = "ACT,
        TRANS = [X=TRANS(VP), IF NUMBERP(X) THEN X-1 ELSE X],
        SEMANTICS = SEMCALL("SEMRS1,SEMANTICS(NP),SEMANTICS(VP));

    FACTORS
        PROB = LK2,
        VOICE = SELECTQ VOICE(VP) WHEN PASS THEN OUT,
        GCASE = IF GCASE(NP) EQUAL "(ACC) THEN OUT ELSE OK,
        MOOD = IF MOOD EQUAL "(WH) THEN GOOD ELSE OK,
        WH = IF MOOD(VP) EQUAL "(WH) THEN POOR ELSE OK,
        NBRAGR = IF GINTERSECT(NBR(NP),NBR(VP)) THEN OK ELSE OUT;

    EXAMPLES
        ONE OF THE SUBMARINES HAS FOUR TORPEDO TUBES (OK)
        SUBMARINES HAS FOUR TORPEDO TUBES (OUT)
        THE SUBMERGED SPEED HAS FOUR TORPEDO TUBES (OUT)
        THE LAFAYETTE HAS FOUR TORPEDO TUBES (OK)
        WHICH SUBMARINE HAS FOUR TORPEDO TUBES (OK)
        NO SUBMARINE HAS MORE THAN TWELVE TORPEDO TUBES (OK)
        THEY HAVE FOUR OF THEM (OK)
        HOW MANY OF THEM HAVE MORE THAN FOUR TORPEDO TUBES (OK);

 END;


RULE.DEF S2     S = NP AUXD VP;

    ATTRIBUTES
        MOOD,FOCUS FROM NP,
        TRANS = [X=TRANS(VP), IF NUMBERP(X) THEN X-1 ELSE X],
        AFFNEG FROM AUXD,
        SEMANTICS = SEMCALL("SEMRS2,SEMANTICS(NP),
            SEMANTICS(VP),AFFNEG(AUXD));

    FACTORS
        NBRAGR = IF GINTERSECT(NBR(NP),NBR(AUXD)) THEN OK
            ELSE OUT,
        PROB = LK4,
        GCASE = IF GCASE(NP) EQUAL "(ACC) THEN OUT ELSE OK,
        VOICE = SELECTQ VOICE(VP) WHEN PASS THEN OUT,
        MOOD = IF MOOD EQUAL "(WH) THEN GOOD ELSE OK,
```

```
        WH = IF MOOD(VP) EQUAL "(WH) THEN POOR ELSE OK,
        STRESS = IF VIRTUAL THEN OK ELSE
            IF AFFNEG EQ "AFF AND STRESS(AUXD) EQ "REDUCED
            THEN POOR ELSE OK;


    EXAMPLES
        THE LAFAYETTE DOES HAVE FUEL (POOR) --
            THIS UTTERANCE MAY BE ACCEPTABLE UNDER CERTAIN
            DISCOURSE CONDITIONS AS IN THE CONTRADICTION OF
            SOMETHING IMPLIED OR STATED PREVIOUSLY
        THE LAFAYETTE DOES NOT HAVE FUEL (OK)
        IT DOES HAVE WHAT (POOR)
        SUBS DO OWNED BY THE US (OUT);

 END;


RULE.DEF S3     S = NP;NP1 AUXB NP;NP2;

    ATTRIBUTES
        MOOD,CMU,RELN,FOCUS FROM NP1,
        AFFNEG FROM AUXB,
        SEMANTICS = SEMCALL("SEMRS3,SEMANTICS(NP1),
            SEMANTICS(NP2),AFFNEG(AUXB)),
        TRANS = 0;

    FACTORS
        NBRAGR1 = IF CMU EQUAL "(UNIT) THEN
            [IF NBR(AUXB) EQUAL "(SG) THEN OK ELSE OUT] ELSE
            IF GINTERSECT(NBR(NP1),NBR(AUXB)) THEN OK ELSE OUT,
        NBRAGR2 = IF CMU(NP2) EQUAL "(UNIT) THEN OK ELSE
            IF GINTERSECT(NBR(NP2),NBR(AUXB)) THEN OK ELSE OUT,
        PROB = LK1,
        FOCUS = IF FOCUS(NP1) EQ "INDEF AND FOCUS(NP2) EQ "DEF
            THEN POOR ELSE OK,
        GCASE1 = IF GCASE(NP1) EQUAL "(ACC) THEN OUT ELSE OK,
        GCASE2 = IF GCASE(NP2) EQUAL "(ACC) THEN OUT ELSE OK,
        MOOD1 = IF MOOD EQUAL "(WH) THEN GOOD ELSE OK,
        MOOD2 = IF MOOD EQUAL "(WH) AND MOOD(NP2) EQUAL "(WH)
            THEN POOR ELSE OK,
        AFFNEG = IF MOOD EQUAL "(WH) AND AFFNEG EQ "NEG THEN BAD
            ELSE OK,
        RELN = IF RELN EQ "T THEN
            IF CMU(NP2) EQUAL "(UNIT) THEN VERYGOOD ELSE OK,
        PERSAGR = IF GINTERSECT(PERS(NP1),PERS(AUXB))
            THEN OK ELSE OUT;

    EXAMPLES
        THE LAFAYETTE IS A SUBMARINE (OK)
        THE LAFAYETTE IS SUBMARINES (OUT)
        A LAFAYETTE IS THE SUBMARINE (POOR)
        THEM ARE SUBMARINES (OUT)
```

```
        WHAT IS THEM (OUT)
        WHAT IS IT (GOOD)
        HOW MANY ARE WHAT (POOR)
        IT AM A LAFAYETTE (OUT)
        WHAT ISN'T THE SURFACE DISPLACEMENT OF THE LAFAYETTE (BAD)
        WHAT IS THE SURFACE DISPLACEMENT (GOOD)
        THE SURFACE DISPLACEMENT IS 7000 TONS (VERYGOOD);

 END;


RULE.DEF S4     S = VP;

    ATTRIBUTES
        FOCUS FROM VP,
        MOOD = "(IMP),
        SEMANTICS = SEMCALL("SEMRS4,SEMANTICS(VP)),

    FACTORS
        WH = IF MOOD(VP) EQUAL "(WH) THEN OUT ELSE OK,
        VOICE = SELECTQ VOICE(VP) WHEN PASS THEN OUT,
        PROB = LK2,
        IMP = SELECTQ IMP(VP) WHEN (YES, UNDEFINED)
            THEN OK ELSE OUT,
        NBR = IF NBR(VP) EQUAL "(SG) THEN OUT ELSE OK;

    EXAMPLES
        LIST WHICH SUBS (OUT)
        LIST SIX SUBS (OK)
        LISTS SIX SUBS (OUT)
        OWNED BY THE RUSSIANS (OUT)
        OWN THE SUBS (OUT);

 END;


RULE.DEF S5     S = AUXD VP;

    ATTRIBUTES
        FOCUS FROM VP,
        MOOD = "(IMP),
        AFFNEG FROM AUXD,
        SEMANTICS = SEMCALL("SEMRS5,SEMANTICS(VP),AFFNEG(AUXD)),
        TRANS = [X=TRANS(VP), IF NUMBERP(X) THEN X-1 ELSE X];

    FACTORS
        VOICE = SELECTQ VOICE(VP) WHEN PASS THEN OUT,
        PROB = LK6,
        IMP = SELECTQ IMP(VP) WHEN (YES, UNDEFINED)
            THEN OK ELSE OUT,
        NBR = IF NBR(AUXD) EQUAL "(SG) THEN OUT ELSE OK,
        WH = IF MOOD(VP) EQUAL "(WH) THEN OUT ELSE OK,
        AFFNEG = SELECTQ AFFNEG WHEN NEG THEN GOOD ELSE POOR;
```

```
    EXAMPLES
        DONT LIST THE DIESELS (GOOD)
        DO LIST THE NUCS (POOR) --
            ASSUMES THE AFFIRMATIVE EMPHATIC FORM
            IS LESS LIKELY AND THAT IF AUXD IS
            PRESENT, IT IS LIKELY TO BE NEGATIVE;


 END;


RULE.DEF S6     S = NP:NP1 AUXD NP:NP2 VP;

    ATTRIBUTES
        FOCUS FROM NP2,
        MOOD = "(WH),
        AFFNEG FROM AUXD,
        SEMANTICS = SEMCALL("SEMRS6,SEMANTICS(NP1),
            SEMANTICS(NP2),SEMANTICS(VP),AFFNEG(AUXD)),
        TRANS = [X=TRANS(VP), IF NUMBERP(X) THEN X-2 ELSE X];

    FACTORS
        MOOD = IF MOOD(NP1) EQUAL "(WH) THEN GOOD
            ELSE IF MOOD(NP1) EQ "UNDEFINED THEN OK ELSE OUT,
        PROB = LK1,
        NBRAGR = IF GINTERSECT(NBR(NP2),NBR(AUXD))
            THEN OK ELSE OUT,
        VOICE = SELECTQ VOICE(VP) WHEN PASS THEN OUT,
        TRANS = [X=TRANS(VP), IF X EQ "UNDEFINED THEN OK
            ELSE IF X LQ 1 THEN OUT ELSE OK],
        GCASE1 = IF GCASE(NP2) EQUAL "(ACC) THEN BAD ELSE OK,
        GCASE2 = IF GCASE(NP1) EQUAL "(NOM) THEN OUT ELSE OK,
        WH = IF MOOD(VP) EQUAL "(WH) THEN OUT ELSE OK,
        MOOD2 = IF MOOD(NP2) EQUAL "(WH) THEN POOR ELSE OK;

    EXAMPLES
        WHAT SUBS DO WE OWN (OK)
        THE SUBS DO WE OWN (OUT)
        WHAT SUBS DO WE OWN MANY SUBMARINES (OUT) --
            THE LAST EXAMPLE SHOWS THE USE OF THE
            TRANS ATTRIBUTE BY FACTORS;

 END;


RULE.DEF S7     S = AUXD NP VP;

    ATTRIBUTES
        FOCUS FROM NP,
        MOOD = "(YN),
        TRANS = [X=TRANS(VP), IF NUMBERP(X) THEN X-1 ELSE X],
        SEMANTICS = SEMCALL("SEMRS7,SEMANTICS(NP),SEMANTICS(VP)),
        AFFNEG FROM AUXD,
```

```
        PITCHC = FINDPITCHC(PLEFT,PRIGHT);

    FACTORS
        VOICE = SELECTQ VOICE(VP) WHEN PASSIVE THEN OUT,
        PROB = LK2,
        WH = IF MOOD(VP) EQUAL "(WH) THEN OUT ELSE OK,
        MOOD = IF MOOD(NP) EQUAL "(WH) THEN BAD ELSE OK,
        GCASE = IF GCASE(NP) EQUAL "(ACC) THEN BAD ELSE OK,
        NBRAGR = IF GINTERSECT(NBR(NP),NBR(AUXD))
            THEN OK ELSE OUT,
        SCORE IF NOT VIRTUAL,
        PITCHC = IF VIRTUAL THEN OK ELSE
            SELECTQ PITCHC WHEN HIRISE THEN GOOD ELSE OK,
        STRESS = IF VIRTUAL THEN OK ELSE
            SELECTQ STRESS(AUXD) WHEN UNREDUCED THEN GOOD ELSE OK;

    EXAMPLES
        DOES IT HAVE TORPEDO TUBES (OK)
        DOES IT HAVE TORPEDO TUBES?? (GOOD) --
            NOTE: ?? INDICATES A PITCH CONTOUR THAT ENDS
            IN A HIGH RISE, WHICH INCREASES THE LIKELIHOOD
            THAT WE ARE ON THE CORRECT PARSING PATH
        DOES WHAT HAVE TORPEDO TUBES (POOR);

 END;


RULE.DEF S8     S = AUXB NP:NP1 NP:NP2;

    ATTRIBUTES
        RELN,CMU,FOCUS FROM NP1,
        MOOD = "(YN),
        TRANS = 0,
        AFFNEG FROM AUXB,
        SEMANTICS = SEMCALL("SEMRS8,SEMANTICS(NP1),
            SEMANTICS(NP2)),
        PITCHC = FINDPITCHC(PLEFT,PRIGHT);

    FACTORS
        GCASE1 = IF GCASE(NP1) EQUAL "(ACC) THEN OUT ELSE OK,
        PROB = LK1,
        GCASE2 = IF GCASE(NP2) EQUAL "(ACC) THEN OUT ELSE OK,
        MOOD1 = IF MOOD(NP1) EQUAL "(WH) THEN BAD ELSE OK,
        MOOD2 = IF MOOD(NP2) EQUAL "(WH) THEN BAD ELSE OK,
        NBRAGR1 = IF CMU EQUAL "(UNIT) THEN
            [IF NBR(AUXB) EQUAL "(SG) THEN OK ELSE OUT] ELSE
            IF GINTERSECT(NBR(NP1),NBR(NP2)) THEN OK ELSE OUT,
        NBRAGR2 = IF CMU(NP2) EQUAL "(UNIT) THEN OK ELSE
            IF GINTERSECT(NBR(NP2),NBR(AUXB)) THEN OK ELSE OUT,
        PERSAGR = IF GINTERSECT(PERS(NP1),PERS(AUXB))
            THEN OK ELSE OUT,
        FOCUS = IF FOCUS(NP1) EQ "INDEF AND FOCUS(NP2) EQ "DEF
            THEN POOR ELSE OK,
```

```
        RELN = IF RELN EQ "T THEN
            IF CMU EQUAL "(UNIT) THEN VERYGOOD ELSE OK,
        SCORE IF NOT VIRTUAL,
        STRESS = IF VIRTUAL THEN OK ELSE
            SELECTQ STRESS(AUXB) WHEN UNREDUCED THEN GOOD,
        PITCHC = IF VIRTUAL THEN OK ELSE
            IF PITCHC EQ "HIRISE THEN GOOD ELSE OK;

    EXAMPLES
        IS A LAFAYETTE THE SUBMARINE (POOR)
        IS IT A LAFAYETTE?? (GOOD) --
            NOTE: ?? INDICATES A PITCH CONTOUR THAT ENDS
            IN A HIGH RISE, WHICH INCREASES THE LIKELIHOOD
            THAT WE ARE ON THE CORRECT PARSING PATH
        IS WHAT THE SURFACE DISPLACEMENT (BAD)
        IS THE LAFAYETTE A SUBMARINE (OK);

 END;


RULE.DEF S9    S = NP AUXB VP;

    ATTRIBUTES
        MOOD,FOCUS FROM NP,
        AFFNEG FROM AUXB,
        VOICE FROM VP,
        SEMANTICS = SEMCALL("SEMRS2,SEMANTICS(NP),
            SEMANTICS(VP),AFFNEG(AUXB)),
        TRANS = [X=TRANS(VP), IF NUMBERP(X) THEN X-1 ELSE X];

    FACTORS
        VOICE = SELECTQ VOICE(VP) WHEN (PASS,UNDEFINED)
            THEN OK ELSE OUT,
        PROB = LK5,
        AGENCY = SELECTQ AGENCY(VP)
            WHEN (YES,UNDEFINED) THEN OK
            ELSE OUT,
        GCASE = IF GCASE(NP) EQUAL "(ACC) THEN OUT ELSE OK,
        MOOD = IF MOOD EQUAL "(WH) THEN GOOD ELSE OK,
        PERSAGR = IF GINTERSECT(PERS(NP),PERS(AUXB))
            THEN OK ELSE OUT,
        NBRAGR = IF GINTERSECT(NBR(NP),NBR(AUXB))
            THEN OK ELSE OUT;

    EXAMPLES
        WHICH IS OWNED BY THE U.S. (OK)
        THAT ONE IS OWNED BY THE U.S. (OK)
        WHICH ONE IS HAD BY THE U.S. (OUT);

 END;
```

```
RULE.DEF S10    S = AUXB NP VP;

    ATTRIBUTES
        AFFNEG FROM AUXB,
        VOICE FROM VP,
        MOOD = "(YN),
        TRANS = [X=TRANS(VP), IF NUMBERP(X) THEN X-1 ELSE X],
        SEMANTICS = SEMCALL("SEMRS10,SEMANTICS(NP),SEMANTICS(VP)),
        PITCHC = FINDPITCHC(PLEFT,PRIGHT);

    FACTORS
        MOOD = IF MOOD(NP) EQUAL "(WH) THEN BAD ELSE OK,
        PROB = LK4,
        VOICE = SELECTQ VOICE(VP) WHEN (PASSIVE,UNDEFINED)
            THEN OK ELSE OUT,
        AGENCY = SELECTQ AGENCY(VP)
            WHEN (YES,UNDEFINED) THEN OK
            ELSE OUT,
        PERSAGR = IF GINTERSECT(PERS(NP),PERS(AUXB))
            THEN OK ELSE OUT,
        NBRAGR = IF GINTERSECT(NBR(NP),NBR(AUXB))
            THEN OK ELSE OUT,
        GCASE = IF GCASE(NP) EQUAL "(ACC) THEN OUT ELSE OK,
        SCORE IF NOT VIRTUAL,
        PITCHC = IF VIRTUAL THEN OK ELSE
        SELECTQ PITCHC WHEN HIRISE THEN GOOD;

    EXAMPLES
        IS IT LISTED (OK)
        IS WHICH LISTED (POOR)
        IS IT LIST (OUT);

 END;


EOF
```

```
    INFIX FILE NPRULE.GRM

    SECTION(71, "(71 72 0));


    %'NOUN PHRASE RULES'


RULE.DEF NP1    NP = NOM;

    ATTRIBUTES
        SEMANTICS FROM NOM,
        FOCUS = "INDEF,
        MOOD = "(DEC),
        SUBCAT,RELN,NBR,CMU FROM NOM;

    FACTORS
        NBRCHK = IF NBR EQUAL "(SG) THEN
        IF GINTERSECT("(MASS),CMU) THEN OK ELSE BAD
        ELSE OK,
        SUBCAT = IF SUBCAT(NOM) EQ "PROPN THEN BAD;

    EXAMPLES
        FUEL (OK AS COMPLETE NP)
        U.S. (BAD AS COMPLETE NP)
        SUBMARINE (BAD AS COMPLETE NP);

 END;


RULE.DEF NP2    NP = NUMBERP;

    ATTRIBUTES
        SEMANTICS = SEMCALL("SEMRNP2,SEMANTICS(NUMBERP),NBR,NUM),
        MOOD,NUM,NBR FROM NUMBERP,
        ELLIPSE = "YES,
        GENSUFF = "NO,
        FOCUS = "INDEF;

    FACTORS
        PROB = LK3,
        HUN = SELECTQ FSTWD(NUMBERP) WHEN HUNDRED THEN OUT;

    EXAMPLES
        HUNDRED (OUT)
        HOW MUCH (OK)
        MORE THAN FOUR (OK);

 END;
```

```
RULE,DEF NP3    NP = NUMBERP "OF NP;

    ATTRIBUTES
        FOCUS = "INDEF,
        SEMANTICS = SEMCALL("SEMRNP3,SEMANTICS(NUMBERP),
            SEMANTICS(NP),NBR(NP),NUM(NUMBERP),NUM(NP)),
        CMU = GINTERSECT(CMU(NUMBERP),CMU(NP)),
        GENSUFF = "NO,
        NUM,NBR,MOOD FROM NUMBERP;

    FACTORS
        FOCUS = SELECTQ FOCUS(NP) WHEN INDEF THEN POOR,
        PROB = LK5,
        NUMCHK = [X=NUM(NP),Y=NUM, IF NUMBERP(X) AND NUMBERP(Y)
            AND X LQ Y THEN BAD ELSE OK],
        CMU = SELECTQ CMU WHEN NIL THEN OUT,
        GCASE = IF GCASE(NP) EQUAL "(NOM) THEN OUT ELSE OK,
        MOOD = IF MOOD(NP) EQUAL "(WH) THEN POOR ELSE OK,
        UNIT = IF "UNIT IN CMU(NP) THEN BAD ELSE OK,
        HUN = SELECTQ FSTWD(NUMBERP) WHEN HUNDRED THEN OUT;

    EXAMPLES
        TWENTY OF SUBMARINES (POOR)
        TWENTY OF THE SUBMARINES (OK)
        MANY OF THE FUEL (OUT)
        FIVE OF THE SPEEDS OF FIVE KNOTS (OUT)
        TWO OF THE SPEEDS OF SUBMARINES (OK)
        HUNDRED OF THE SUBMARINES (OUT);
 END;


RULE,DEF NP4    NP = NUMBERP NOM;

    ATTRIBUTES
        FOCUS = "INDEF,
        MOOD,NUM FROM NUMBERP,
        NBR = GINTERSECT(NBR(NUMBERP),NBR(NOM)),
        RELN FROM NOM,
        SEMANTICS = SEMCALL("SEMRNP4,SEMANTICS(NUMBERP),
            SEMANTICS(NOM),NBR(NOM),CMU(NOM),MOOD),
        CMU = GINTERSECT(CMU(NUMBERP),CMU(NOM));

    FACTORS
        CMU = SELECTQ CMU WHEN NIL THEN OUT,
        PROB = LK1,
        HUN = IF FSTWD(NUMBERP) IN "(HUNDRED THOUSAND MILLION)
            THEN OUT,
        NBR = SELECTQ NBR WHEN NIL THEN OUT,
        UNIT = IF "UNIT IN CMU THEN VERYGOOD ELSE OK,
        RELN = IF RELN EQ T THEN OUT ELSE OK,
        SUBCAT = SELECTQ SUBCAT(NOM) WHEN PROPN THEN OUT;
```

```
    EXAMPLES
        FIVE FUELS (OUT)
        HOW MUCH SUBMARINE (OUT)
        ONE SUBMARINES (OUT)
        HOW MANY FUEL (OUT)
        FIVE FEET (VERYGOOD)
        FIVE SUBMERGED SPEEDS OF THREE KNOTS (OUT)
        FIVE SUBMERGED SPEEDS OF THE SUBS (OUT)
        FIVE SUBMARINES (OK);

 END;


RULE.DEF NP5    NP = DET;

    ATTRIBUTES
        ELLIPSE = "YES,
            GENSUFF = "NO,
        SEMANTICS = SEMCALL("SEMRNP6,NBR(DET),MOOD,FOCUS),
        MOOD,NBR,FOCUS FROM DET;

    FACTORS
        PROB = LK1;

    EXAMPLES
        WHICH (OK)
        THOSE (OK);

 END;


RULE.DEF NP6    NP = DET "OF NP;

    ATTRIBUTES
        SEMANTICS = SEMCALL("SEMRNP7,FOCUS(DET),SEMANTICS(NP)),
        GENSUFF = "NO,
        MOOD,NBR,FOCUS FROM DET;

    FACTORS
        PROB = LK5,
        FOCUS = SELECTQ FOCUS(NP) WHEN INDEF THEN BAD,
        GCASE = IF GCASE(NP) EQUAL "(NOM) THEN OUT ELSE OK,
        MOOD = IF MOOD(NP) EQUAL "(WH) THEN POOR ELSE OK,
        UNIT = IF CMU(NP) EQUAL "(UNIT) THEN BAD ELSE OK,
        STRCHK = IF STRING(DET)  EQUAL "(WHICH) THEN GOOD ELSE OK;

    EXAMPLE
        WHICH OF THEM (OK)
        WHICH OF THE KNOTS (OUT);

 END;
```

```
RULE.DEF NP7    NP = DET NOM;

    ATTRIBUTES
        FOCUS = "DEF,
        CMU = GINTERSECT(CMU(DET),CMU(NOM)),
        SEMANTICS = SEMCALL("SEMRNP8,SEMANTICS(NOM),GCASE(DET),
            MOOD,FOCUS),
        NBR = GINTERSECT(NBR(DET),NBR(NOM)),
        RELN FROM NOM,
        MOOD FROM DET;

    FACTORS
        CMUCHK = SELECTQ CMU WHEN NIL THEN OUT ELSE OK,
        PROB = LK2,
        UNIT = IF "UNIT IN CMU THEN POOR ELSE OK,
        NBRCHK = SELECTQ NBR WHEN NIL THEN OUT;

    EXAMPLES
        THOSE SUBMARINE (OUT)
        THAT SUBMARINE (OK)
        THOSE FUELS (OUT)
        THAT FUEL (OK)
        WHICH TONS (POOR)
        THAT DRAFT OF FIVE FEET (POOR)
        WHAT FUEL (OK)
        WHICH SUBMARINE (OK)
        THAT SPEED (OK)
        THAT SURFACE DISPLACEMENT (OK);

 END;


RULE.DEF NP8    NP = DET NUMBER "OF NP;

    ATTRIBUTES
        MOOD,FOCUS FROM DET,
        SEMANTICS = SEMCALL("SEMRNP9,FOCUS,GCASE(DET),
            SEMANTICS(NUMBER),SEMANTICS(NP),NBR(NUMBER),
            NUM(NUMBER),NUM(NP)),
        NBR = GINTERSECT(NBR(DET),NBR(NUMBER)),
        NUM FROM NUMBER,
        CMU FROM NP,
        GENSUFF = "NO;

    FACTORS
        NBR = IF NBR(NP) EQUAL "(SG) OR NBR EQ NIL THEN OUT
            ELSE OK,
        PROB = LK6,
        FOCUS = SELECTQ FOCUS(NP) WHEN INDEF THEN BAD,
        GCASE = IF GCASE(NP) EQUAL "(NOM) THEN OUT ELSE OK,
        UNIT = IF "UNIT IN CMU(NP) THEN BAD ELSE OK,
        MASS = IF CMU EQUAL "(MASS) THEN OUT ELSE OK,
        MOOD = IF MOOD(NP) EQUAL "(WH) THEN POOR ELSE OK;
```

```
    EXAMPLES
        THOSE TWO OF THE KNOTS (BAD)
        THESE TWO OF THE SIX (OK)
        WHICH TWO OF THE SPEEDS OF FIVE KNOTS (BAD)
        WHICH TWO OF THE SPEEDS OF SUBMARINES (OK)
        THOSE TWO OF SOME SUBS (BAD);

 END;


RULE.DEF NP9    NP = DET NUMBER NOM;

    ATTRIBUTES
        NBR = GINTERSECT(NBR(NOM),
            GINTERSECT(NBR(DET),NBR(NUMBER))),
        MOOD,FOCUS FROM DET,
        SEMANTICS = SEMCALL("SEMRNP10,FOCUS,SEMANTICS(NUMBER),
            SEMANTICS(NOM),GCASE(DET)),
        NUM FROM NUMBER,
        CMU FROM NOM,
        RELN FROM NOM;

    FACTORS
        PROB = LK4,
        UNIT = IF "UNIT IN CMU(NOM) THEN BAD ELSE OK,
        MASS = IF CMU EQUAL "(MASS) THEN OUT ELSE OK,
        NBR = SELECTQ NBR WHEN NIL THEN OUT;

    EXAMPLES
        WHICH FIVE TONS (BAD)
        THOSE FIVE SUBS (OK)
        WHICH TWO SPEEDS OF FIVE KNOTS (BAD)
        WHICH TWO SPEEDS OF SUBMARINES (OK)
        THAT ONE FUEL (OUT);

 END;


RULE.DEF NP10    NP = DET NUMBER;

    ATTRIBUTES
        NBR = GINTERSECT(NBR(DET),NBR(NUMBER)),
        SEMANTICS = SEMCALL("SEMRNP11,FOCUS,SEMANTICS(NUMBER),
        GCASE(DET)),
        GENSUFF = [X=STRING(NUMBER), IF X EQUAL "(ONE) THEN "YES
            ELSE IF X EQ "UNDEFINED THEN "UNDEFINED ELSE "NO],
        MOOD,FOCUS FROM DET;

    FACTORS
        PROB = LK4,
        NBR = SELECTQ NBR WHEN NIL THEN OUT;
```

```
    EXAMPLES
        THAT HUNDRED (OK)
        THIS ONE (OK);

 END;


RULE.DEF NP11     NP = ART NOM;

    ATTRIBUTES
        RELN FROM NOM,
        CMU = GINTERSECT(CMU(ART),CMU(NOM)),
        SEMANTICS = SEMCALL("SEMRNP12,SEMANTICS(NOM),MOOD,FOCUS),
        NBR = GINTERSECT(NBR(ART),NBR(NOM)),
        MOOD = "DEC,
        FOCUS FROM ART;

    FACTORS
        CMU = SELECTQ CMU WHEN NIL THEN OUT,
        PROB = LK1,
        NBR = SELECTQ NBR WHEN NIL THEN OUT,
        UNIT = IF "UNIT IN CMU THEN IF FOCUS EQ "DEF
            THEN POOR ELSE GOOD,
        RELN = IF RELN EQ T AND IF FOCUS EQ "INDEF AND
            IF CMU EQ "(COUNT) THEN OUT ELSE OK,
        PROPNCHK = IF SUBCAT(NOM) EQ "PROPN THEN
            [X=FSTWD(ART), IF X EQ "THE THEN GOOD
            ELSE IF X EQ "UNDEFINED THEN OK ELSE OUT]
            ELSE OK;

    EXAMPLES
        A SUBMARINES (OUT)
        THE TON (POOR)
        THE DRAFT OF FIVE FEET (POOR)
        A FUEL (OUT)
        THE SUBMARINE (OK)
        A TON (GOOD)
        A SUBMARINE (OK)
        A DRAFT OF FIVE FEET (GOOD)
        THE SUBMERGED SPEED (OK)
        A DRAFT OF THE LAFAYETTE (OUT);

 END;


RULE.DEF NP12     NP = ART NUMBER "OF NP;

    ATTRIBUTES
        MOOD,FOCUS FROM ART,
        SEMANTICS = SEMCALL("SEMRNP9,FOCUS,NIL,SEMANTICS(NUMBER),
            SEMANTICS(NP),NBR(NUMBER),NUM(NUMBER),NUM(NP)),
        NBR = GINTERSECT(NBR(ART),NBR(NUMBER)),
        GENSUFF = "NO,
```

```
        CMU = "(COUNT);

    FACTORS
        ACHK = IF FSTWD(ART) EQ "A
             THEN [Y=FSTWD(NUMBER),
             IF Y NQ "HUNDRED OR Y NQ "UNDEFINED THEN OUT ELSE OK]
             ELSE OK,
        PROB = LK6,
        GCASE = IF GCASE(NP) EQUAL "(NOM) THEN OUT ELSE OK,
        MOOD = IF MOOD(NP) EQUAL "(WH) THEN POOR ELSE OK,
        UNIT = IF "UNIT IN CMU(NP) THEN BAD ELSE OK,
        FOCUS = SELECTQ FOCUS(NP) WHEN INDEF THEN BAD;

    EXAMPLES
        A HUNDRED OF THE TONS (BAD)
        A ONE HUNDRED OF THE SUBS (OUT)
        THE FIVE OF THE SPEEDS OF FIVE KNOTS (BAD)
        THE FIVE OF THE SPEEDS OF THE SUBS (OK)
        A HUNDRED OF THE SUBS (OK);

END;


RULE.DEF NP13    NP = ART NUMBER NOM;

    ATTRIBUTES
        NBR = GINTERSECT(NBR(NOM),NBR(NUMBER)),
        MOOD,FOCUS FROM ART,
        SEMANTICS = SEMCALL("SEMRNP10,FOCUS,
        SEMANTICS(NUMBER),SEMANTICS(NOM)),
        CMU,RELN FROM NOM;

    FACTORS
        PROB = LK4,
        UNIT = IF CMU EQUAL "(UNIT) THEN
             IF FOCUS EQ "INDEF THEN GOOD ELSE POOR
             ELSE OK,
        NBR = SELECTQ NBR WHEN NIL THEN OUT,
        ACHK = IF FSTWD(ART) EQ "A THEN [Y=FSTWD(NUMBER),
             IF Y NQ "HUNDRED OR Y NQ "UNDEFINED THEN OUT ELSE OK]
             ELSE OK;

    EXAMPLES
        A HUNDRED TONS (GOOD)
        THE FIVE SUBS (OK)
        THE HUNDRED TONS (POOR);

END;


RULE.DEF NP14    NP = ART NUMBER;
```

```
    ATTRIBUTES
        NBR = GINTERSECT(NBR(ART),NBR(NUMBER)),
        SEMANTICS = SEMCALL("SEMRNP11,FOCUS,SEMANTICS(NUMBER)),
        MOOD,FOCUS FROM ART;

    FACTORS
        PROB = LK4,
        ACHK = IF FSTWD(ART) EQ "A THEN [Y=FSTWD(NUMBER),
            IF Y NQ "HUNDRED OR Y NQ "UNDEFINED
            THEN OUT ELSE OK];

    EXAMPLES
        A FIVE (OUT)
        THE FIVE (OK)
        A HUNDRED (OK);

 END;


EOF
```

```
    INFIX FILE NRULES.GRM

    SECTION(71, "(71 72 0 72));

    %'NOUN AND NOMHEAD RULES'

RULE.DEF NOM1    NOM = NOMHEAD;

    ATTRIBUTES
        SEMANTICS FROM NOMHEAD,
        CMU,SUBCAT,NBR,RELN FROM NOMHEAD;

    EXAMPLES
        SUBMERGED SPEED (OK);

 END;

RULE.DEF NH1    NOMHEAD = NOMHEAD PREPP;

    ATTRIBUTES
        CMU = IF RELN EQ T THEN
            GUNION(CMU(NOMHEAD),CMU(PREPP)) ELSE CMU(NOMHEAD),
        RELN,SUBCAT,NBR FROM NOMHEAD,
        SEMANTICS = SEMCALL("SEMRNH1,SEMANTICS(NOMHEAD),
            SEMPREP(PREPP),SEMANTICS(PREPP));

    FACTORS
        PROB = LK1,
        FSTWD = IF FSTWD(PREPP) EQ "OF THEN GOOD ELSE OK,
        MOOD = IF MOOD(PREPP) EQ "(WH) THEN POOR ELSE OK,
        RELN = IF RELN THEN VERYGOOD ELSE OK;

    EXAMPLES
        SPEED OF WHAT (POOR)
        SPEED OF TWENTY KNOTS (VERYGOOD);

 END;

RULE.DEF NH2    NOMHEAD = NOUN;

    ATTRIBUTES
        SEMANTICS = SEMCALL("SEMRNH2,SEMANTICS(NOUN),NBR(NOUN)),
        NBR,RELN,CMU,SUBCAT FROM NOUN;

    FACTORS
        PROB = LK1;

    EXAMPLES
        FEET (OK)
```

```
        LAFAYETTES (OK)
        SPEED (OK);

 END;


RULE.DEF N1    NOUN = N;

    ATTRIBUTES
        CMU,RELN,SUBCAT FROM N,
        SEMANTICS  FROM N,
        NBR = "(SG);

    FACTORS
        PLSUFF = IF PLSUFF(N) EQ "NO THEN GOOD ELSE OK,
        RELN = IF RELN EQ "T THEN GOOD ELSE OK,
        CMU = IF "MASS IN CMU THEN GOOD ELSE OK,
        PROB = LK2;
        %'THIS MAKES THE PL BE TRIED FIRST
        AND FORCES THE LONGEST MATCH.'

    EXAMPLES

        FUEL (GOOD)
        FOOT (GOOD)
        TORPEDO TUBE (OK)
        SUBMARINE (OK);

 END;


RULE.DEF N2    NOUN = N -PL;

    ATTRIBUTES
        CMU,RELN,SUBCAT FROM N,
        SEMANTICS FROM N,
        MAPINFO = MAPSUFFIX(LEFT(N),RIGHT(N),SPELLING(N),"PL,"S),
        RIGHT = [X=MAPINFO, IF X NQ "UNDEFINED THEN CADR(X)
            ELSE "UNDEFINED],
        STRING = [X=SPELLING(N),IF X EQ "UNDEFINED THEN "(NIL PL)
            ELSE LIST(X,"PL)],
        NBR = "(PL);

    FACTORS
        PLSUFF = IF PLSUFF(N) EQ "NO THEN OUT,
        PROB = LK1,
        CMU = IF CMU EQUAL "(MASS) THEN OUT ELSE OK,
        RELN = IF RELN EQ "T THEN POOR ELSE OK,
        SCORE,
        MAPI = IF VIRTUAL THEN OK ELSE
            [X=MAPINFO, IF X EQ "UNDEFINED THEN OK
            ELSE CADDR(X)];
```

```
    EXAMPLES
        FOOT -S (OUT)
        SURFACE.DISPLACEMENT -S (POOR)
        FUEL -S (OUT)
        TON -S (GOOD)
        SUBMARINE -S (OK);

 END;


EOF
```

```
    INFIX FILE VPRULE.GRM

    SECTION(71, "(71 72 0));


    %'VERB RULES'


RULE.DEF VP1     VP = VERB;

    ATTRIBUTES
        SEMANTICS = SEMCALL("SEMRVP1,SEMANTICS(VERB),VOICE(VERB),
        FSTWD(VERB)),
        AGENCY,IMP,NBR,VOICE,TRANS FROM VERB;

    FACTORS
        PROB = LK1;

    EXAMPLES
        LIST (OK);

 END;


RULE.DEF VP2     VP = VP NP;

    ATTRIBUTES
        FOCUS FROM NP,
        SEMANTICS = SEMCALL("SEMRVP2,SEMANTICS(VP),
        SEMANTICS(NP)),
        MOOD = IF MOOD(NP) EQUAL "(WH) THEN "(WH)
            ELSE MOOD(VP),
        AGENCY,IMP,NBR,VOICE FROM VP,
        TRANS = [X=TRANS(VP),IF NUMBERP(X) THEN X-1 ELSE X];

    FACTORS
        TRANS = IF TRANS EQ 0 THEN BAD,
        PROB = LK1,
        MOOD = IF MOOD EQUAL "(WH) THEN POOR ELSE OK,
        GCASE = IF GCASE(NP) EQUAL "(NOM) THEN BAD ELSE OK;

    EXAMPLES
        LIST THEM (OK);

 END;


RULE.DEF VP3     VP = VP PREPP;

     ATTRIBUTES
        AGENCY,IMP,NBR,VOICE FROM VP,
        FOCUS FROM PREPP,
        MOOD = IF MOOD(PREPP) EQU   "(WH) THEN "(WH)
```

```
            ELSE MOOD(VP),
        SEMANTICS = SEMCALL("SEMRVP3,SEMANTICS(VP),
        SEMANTICS(PREPP),SEMPREP(PREPP)),
        TRANS = [X=TRANS(VP), IF NUMBERP(X) THEN X-1 ELSE X];

    FACTORS
        TRANS = IF TRANS EQ 0 THEN BAD,
        PROB = LK3,
        MOOD = IF MOOD EQUAL "(WH) THEN POOR ELSE OK;

    EXAMPLES
        OWNED BY THE RUSSIANS (OK);

 END;


RULE.DEF V1    VERB = V;

    ATTRIBUTES
        NBR = "(PL),
        SEMANTICS FROM V,
        AGENCY,IMP,TRANS FROM V;

    FACTORS
        PROB = LK1;

    EXAMPLES
        LIST (OK);

 END;


RULE.DEF V2    VERB = V -SG;

    ATTRIBUTES
        SEMANTICS FROM V,
        NBR = "(SG),
        MAPINFO = MAPSUFFIX(LEFT(V),RIGHT(V),SPELLING(V),"SG,"S),
        RIGHT = [X=MAPINFO,
            IF X EQ "UNDEFINED THEN X ELSE CADR(X)],
        STRING = [X=SPELLING(V), IF X EQ "UNDEFINED THEN "(NIL SG)
            ELSE LIST(X,"SG)],
        AGENCY,IMP,TRANS FROM V;

    FACTORS
        MAPI = IF VIRTUAL THEN OK ELSE
            [X=MAPINFO, IF X EQ "UNDEFINED THEN OK ELSE CADDR(X)],
        PROB = LK2;

    EXAMPLES
        LIST (OK)
        LISTS (OK);
```

```
END;


RULE.DEF V3     VERB = V -PPL;

    ATTRIBUTES
        SEMANTICS FROM V,
        VOICE = "PASSIVE,
        MAPINFO = MAPSUFFIX(LEFT(V),RIGHT(V),SPELLING(V),
            "PPL,"ED),
        RIGHT = [X=MAPINFO,
            IF X EQ "UNDEFINED THEN X ELSE CADR(X)],
        MAPINFO = MAPSUFFIX(LEFT(DO),RIGHT(DO),SPELLING(DO),
            "NT,"NT),
        STRING = [X=SPELLING(V), IF X EQ "UNDEFINED
            THEN "(NIL ED)
            ELSE LIST(X,"ED)],
        TRANS FROM V;

    FACTORS
        PROB = LK3,
        AGENCY = SELECTQ AGENCY(V) WHEN NO THEN OUT,
        MAPI = IF VIRTUAL THEN OK ELSE
            [X=MAPINFO, IF X EQ "UNDEFINED THEN OK ELSE CADDR(X)],
        TRANS = [X=TRANS(V), IF X EQ "UNDEFINED THEN OK
            ELSE IF X LQ 1 THEN OUT ELSE OK];

    EXAMPLES
        OWNED (OK)
        LISTED (OK)
        (THIS WILL HAVE TO BE CHANGED WHEN PAST TENSE AND
        PERFECT ASPECT ARE ADDED; E.G.,  WE OWNED IT,
        WE HAVE OWNED IT);

END;


EOF
```

```
    INFIX FILE AUXRUL.GRM

    SECTION(71, "(71 72 0));


    %'AUXILIARY RULES'


RULE.DEF D1    AUXD = DO;

    ATTRIBUTES
        NBR,STRESS FROM DO,
        SEMANTICS FROM DO;

    FACTORS
        PROB = LK1;

    EXAMPLES
        DO (OK)
        DOES (OK)
        DON'T (OK);

 END;


RULE.DEF D2    AUXD = DO NEG;

    ATTRIBUTES
        NBR FROM DO,
        AFFNEG = "NEG,
        SEMANTICS FROM DO,
        STRESS = MAXSTRESS(STRESS(DO),STRESS(NEG));

    FACTORS
        PROB = LK2;

    EXAMPLES
        DO NOT (OK);

 END;


RULE.DEF D3    AUXD = DO -NT;

    ATTRIBUTES
        NBR,STRESS FROM DO,
        SEMANTICS FROM DO,
        RIGHT = [X=MAPINFO,
            IF X EQ "UNDEFINED THEN X ELSE CADR(X)],
        MAPINFO = MAPSUFFIX(LEFT(DO),
            RIGHT(DO),SPELLING(DO),"NT,"NT),
        STRING = [X=SPELLING(DO), IF X EQ "UNDEFINED
            THEN "(NIL NT)
```

```
            ELSE LIST(X,"NT)],
        AFFNEG = "NEG;

    FACTORS
        PROB = LK2,
        MAPI = IF VIRTUAL THEN OK ELSE
            [X=MAPINFO, IF X EQ "UNDEFINED THEN OK ELSE CADDR(X)],
        STRESS = IF VIRTUAL THEN OK ELSE
            SELECTQ STRESS(DO) WHEN REDUCED THEN OUT;

    EXAMPLES
        DOES (OK)
        DOESN'T (OK);

 END;


RULE.DEF B1     AUXB = BE;

    ATTRIBUTES
        SEMANTICS FROM BE,
        NBR,PERS,STRESS FROM BE;

    FACTORS
        PROB = LK1;

    EXAMPLES
        IS (OK)
        ARE (OK)
        AM (OK);

 END;


RULE.DEF B2     AUXB = BE NEG;

    ATTRIBUTES
        NBR,PERS FROM BE,
        AFFNEG = "NEG,
        SEMANTICS FROM BE,
        STRESS = MAXSTRESS(STRESS(BE),STRESS(NEG));

    FACTORS
        PROB = LK2,
        STRESS = IF VIRTUAL THEN OK ELSE
            SELECTQ STRESS(NEG) WHEN REDUCED THEN POOR;

    EXAMPLES
        IS NOT (OK)
        ARE NOT (OK)
        AM NOT (OK)
```

```
 END;


RULE,DEF B3    AUXB = BE -NT;

    ATTRIBUTES
        SEMANTICS FROM BE,
        NBR,STRESS,PERS FROM BE,
        MAPINFO = MAPSUFFIX(LEFT(BE),RIGHT(BE),SPELLING(BE),
            "NT,"NT),
        RIGHT = [X=MAPINFO,
            IF X EQ "UNDEFINED THEN X ELSE CADR(X)],
        STRING = [X=SPELLING(BE), IF X EQ "UNDEFINED
            THEN "(NIL NT)
            ELSE LIST(X,"NT)],
        AFFNEG = "NEG;

    FACTORS
        PROB = LK2,
        STRESS = IF VIRTUAL THEN OK ELSE
            SELECTQ STRESS(BE) WHEN REDUCED THEN POOR,
        MAPI = IF VIRTUAL THEN OK ELSE
            [X=MAPINFO, IF X EQ "UNDEFINED THEN OK ELSE CADDR(X)],
        SGCHK = IF NBR(BE) EQUAL "(SG) AND PERS(BE) EQ 1
            THEN BAD ELSE OK;

    EXAMPLES
        ISN'T (OK)
        AREN'T (OK);

 END;


EOF
```

```
    INFIX FILE MIRULE.GRM

    SECTION(71, "(71 72 0));


    %'MISC RULES'


RULE.DEF PREPP1    PREPP = PREP NP;

    ATTRIBUTES
        SEMANTICS FROM NP,
        SEMPREP FROM PREP,
        FOCUS,CMU,NBR,RELN,MOOD FROM NP;

    FACTORS
        GCASE = IF GCASE(NP) EQUAL "(NOM) THEN OUT ELSE OK;

    EXAMPLES
        OF THE LAFAYETTE (OK)
        OF 7000 TONS (OK)
        FOR WHICH SUB (OK)
        BY THE RUSSIANS (OK)
        OF THEY (OUT);

 END;


RULE.DEF DET1    DET = NP -GEN;

    ATTRIBUTES
        GCASE = "(GEN),
        MOOD,SUBCAT,FOCUS FROM NP,
        SEMANTICS = SEMCALL("SEMRTHP1,SEMANTICS(NP),"GEN),
        MAPINFO = MAPSUFFIX(LEFT(NP),RIGHT(NP),SPELLING(NP),
            "GEN,"S),
        RIGHT = [X=MAPINFO,
            IF X EQ "UNDEFINED THEN X ELSE CADR(X)],
        NBR = "(SG PL);

    FACTORS
        GENSUFF = IF GENSUFF(NP) EQ "NO THEN OUT,
        MAPI = IF VIRTUAL THEN OK ELSE
            [X=MAPINFO, IF X EQ "UNDEFINED THEN OK ELSE CADDR(X)],
        SUBCAT = IF SUBCAT EQ "PRO THEN OUT,
        RELN = IF RELN(NP) EQ T THEN BAD ELSE OK,
        CMU = IF CMU(NP) EQUAL "(UNIT) THEN BAD ELSE OK;

    EXAMPLES
        THAT ONE 'S (OK)
        THE LAFAYETTE 'S (OK)
        THE 7000 TONS 'S (BAD)
        THE SURFACE DISPLACEMENT 'S (BAD)
```

```
        WE 'S (OUT);

 END;

EOF
```

```
    INFIX FILE NMPRUL.GRM

    SECTION(71, "(71 72 0));


    %'NUMBERP RULES'


%'DELETE THIS RULE FOR NOW AND HAVE "HOW,MANY" AS SINGLE WORD.
RULE.DEF NUMP1    NUMBERP = "HOW MP;

    ATTRIBUTES
        SEMANTICS = SEMCALL("SEMRNUMP1,"HOW,SEMANTICS(MP)),
        MOOD = "(WH),
        CMU,NBR FROM MP;

    FACTORS
        PROB = LK2,
        STR = [X=FSTWD(MP),
            IF X EQ "UNDEFINED THEN OK
        ELSE
        IF X IN "(MANY MUCH) THEN OK
        ELSE BAD];

    EXAMPLES
        HOW MANY (OK)
        HOW MUCH (OK);

 END;
END OF COMMENT FOR DELETING RULE.'


RULE.DEF NUMP2    NUMBERP = MP;

    ATTRIBUTES
        SEMANTICS FROM MP,
        MOOD,NUM,CMU,NBR FROM MP;

    FACTORS
        PROB = LK2;

    EXAMPLES
        MANY (OK)
        MUCH (OK);

 END;


RULE.DEF NUMP3    NUMBERP = THANR "THAN NUMBER;

    ATTRIBUTES
        NUM = COMBNUM(REL(THANR),NUM(NUMBER)),
        CMU = "(COUNT UNIT),
```

```
        SEMANTICS = SEMCALL("SEMRNUMP3,NUM,SEMANTICS(NUMBER)),
        MOOD = "(DEC),
        NBR FROM NUMBER;

    FACTORS
        PROB = LK1;

    EXAMPLES
        MORE THAN FOUR (OK);

 END;


RULE.DEF NUMP4     NUMBERP = NUMBER;

    ATTRIBUTES
        CMU = "(COUNT UNIT),
        SEMANTICS FROM NUMBER,
        MOOD = "(DEC),
        NUM FROM NUMBER,
        NBR FROM NUMBER;

    FACTORS
        PROB = LK1;

    EXAMPLES
        FOUR (OK);

 END;


EOF
```

```
    INFIX FILE NUMRUL.GRM

    SECTION(71, "(71 72 0 73));


    %'NUMBER RULES'


RULE.DEF NUM1     NUMBER = SMALLNUM;

    ATTRIBUTES
        NUM FROM SMALLNUM,
        SEMANTICS = SEMCALL("SEMRNUMBER,NUM),
        NBR FROM SMALLNUM;

    FACTORS
        PROB = LK3;

    EXAMPLES
        ONE (OK)
        FIFTY ONE (OK)
        TEN (OK);

 END;


RULE.DEF NUM2     NUMBER = BIGADD;

    ATTRIBUTES
        SEMANTICS = SEMCALL("SEMRNUMBER,NUM),
        NUM FROM BIGADD,
        NBR = "(PL);

    FACTORS
        PROB = LK4;

    EXAMPLES
        FOUR HUNDRED AND FIFTY ONE (OK);

 END;


RULE.DEF NUM3     NUMBER = BIGMULT;

    ATTRIBUTES
        SEMANTICS = SEMCALL("SEMRNUMBER,NUM),
        NUM FROM BIGMULT,
        NBR = "(PL);

    FACTORS
        PROB = LK4;
```

```
    EXAMPLES
        FIFTY ONE THOUSAND (OK);

 END;


RULE.DEF NUM4    BIGMULT = SMALLNUM BIGCAT;

    ATTRIBUTES
        NUM = SMULT(NUM(SMALLNUM),NUM(BIGCAT));

    FACTORS
        NUMTYP = IF NUMTYP(SMALLNUM) EQ "DECADE AND
            FSTWD(BIGCAT) EQ "HUNDRED THEN POOR;

    EXAMPLES
        FIFTY ONE THOUSAND (OK);

 END;


RULE.DEF NUM5    BIGMULT = BIGADD BIGCAT;

    ATTRIBUTES
        NUM = SMULT(NUM(BIGADD),NUM(BIGCAT));

    FACTORS
        NUM = [X = NUM(BIGADD),Y=NUM(BIGCAT),
            IF NUMBERP(X) AND NUMBERP(Y) THEN
            IF X LS Y THEN OK ELSE OUT ELSE OK];

    EXAMPLES
        FOUR HUNDRED AND FIFTY ONE THOUSAND (OK);

 END;


RULE.DEF NUM6    BIGMULT = BIGCAT;

    ATTRIBUTES
        NUM FROM BIGCAT;

    EXAMPLES
        HUNDRED (OK)
        THOUSAND (OK);

 END;


RULE.DEF NUM7    SMALLNUM = DIGIT;

    ATTRIBUTES
        NBR = [X=STRING(DIGIT),
```

```
            IF X EQ "UNDEFINED THEN "UNDEFINED
            ELSE
            IF X EQUAL "(ONE) THEN "(SG)
            ELSE "(PL)],
        NUM FROM DIGIT,
        NUMTYP = "DIGIT;

    FACTORS
        DIGTYP = [X=DIGTYP(DIGIT),
            IF X EQ "UNDEFINED OR 1 IN X THEN OK ELSE BAD];

    EXAMPLES
        ONE (OK)
        FIVE (OK)
        FIF (OK)
        NINE (OK);

 END;


RULE.DEF NUM8    SMALLNUM = TEEN;

    ATTRIBUTES
        NBR = "(PL),
        NUM FROM TEEN,
        NUMTYP = "TEEN;

    EXAMPLES
        FIFTEEN (OK)
        NINETEEN (OK);

 END;


RULE.DEF NUM9    TEEN = DIGIT -TEEN;

    ATTRIBUTES
        NUM = SADD(NUM(DIGIT),10),
        MAPINFO = MAPSUFFIX(LEFT(DIGIT),RIGHT(DIGIT),
            SPELLING(DIGIT),"TEEN,"TEEN),
        STRING = [X=SPELLING(DIGIT), IF X EQ "UNDEFINED THEN
            "(NIL TEEN) ELSE LIST(X,"TEEN)],
        RIGHT = [X=MAPINFO,
            IF X NQ "UNDEFINED THEN CADR(X) ELSE "UNDEFINED],
        NBR = "(PL);

    FACTORS
        MAPI = IF VIRTUAL THEN OK ELSE
            [X=MAPINFO, IF X EQ "UNDEFINED THEN OK ELSE CADDR(X)],
        DIGTYP = [X=DIGTYP(DIGIT),
            IF X EQ "UNDEFINED OR 2 IN X THEN OK ELSE BAD];
```

```
    EXAMPLES
        FIFTEEN (OK)
        NINETEEN (OK);

 END;


RULE.DEF NUM10    DIGTY = DIGIT -TY;

    ATTRIBUTES
        NUM = SMULT(NUM(DIGIT),10),
        MAPINFO = MAPSUFFIX(LEFT(DIGIT),RIGHT(DIGIT),
            SPELLING(DIGIT),
            "TY,"TY),
        RIGHT = [X=MAPINFO,
            IF X NQ "UNDEFINED THEN CADR(X) ELSE "UNDEFINED],
        STRING = [X=SPELLING(DIGIT), IF X EQ "UNDEFINED THEN
            "(NIL TY) ELSE LIST(X,"TY)],
        NBR = "(PL);

    FACTORS
        DIGTYP = [X=DIGTYP(DIGIT),
            IF X EQ "UNDEFINED OR 3 IN X THEN OK ELSE BAD],
        SCORE,
        MAPI = IF VIRTUAL THEN OK ELSE
            [X=MAPINFO, IF X EQ "UNDEFINED THEN OK
            ELSE CADDR(X)];

    EXAMPLES
        FIFTY (OK)
        NINETY (OK);

 END;


RULE.DEF NUM11    SMALLNUM = DIGTY;

    ATTRIBUTES
        NUM FROM DIGTY,
        NBR = "(PL),
        NUMTYP = "DECADE;

    EXAMPLES
        FIFTY (OK)
        NINETY (OK);

 END;


RULE.DEF NUM12    SMALLNUM = DIGTY DIGIT;

    ATTRIBUTES
        NBR = "(PL),
```

```
        NUM = SADD(NUM(DIGTY),NUM(DIGIT)),
        NUMTYP = "DECADEPLUS;

    FACTORS
        PROB = LK1,
        DIGTYP = [X=DIGTYP(DIGIT),
            IF X EQ "UNDEFINED OR 1 IN X THEN OK ELSE BAD];

    EXAMPLES
        FIFTY TWO (OK);

 END;


RULE.DEF NUM13     BIGADD = BIGMULT SMALLNUM;

    ATTRIBUTES
        NUM = SADD(NUM(BIGMULT),NUM(SMALLNUM));

    EXAMPLES
        TWO HUNDRED THOUSAND FORTY SEVEN (OK)
        TWO HUNDRED THOUSAND TWO FORTY SEVEN (OUT);

 END;


RULE.DEF NUM14     BIGADD = BIGMULT "AND SMALLNUM;

    ATTRIBUTES
        NUM = SADD(NUM(BIGMULT),NUM(SMALLNUM));

    EXAMPLES
        FOUR HUNDRED AND FIFTY TWO (OK);

 END;


RULE.DEF NUM15     BIGADD = BIGMULT BIGADD;

    ATTRIBUTES
        NUM = SADD(NUM(BIGMULT),NUM(BIGADD));

    FACTORS
        NUM = [X=NUM(BIGMULT),Y=NUM(BIGADD), IF NUMBERP(X) AND
            NUMBERP(Y) THEN IF X GR Y THEN OK ELSE OUT ELSE OK];
        %'THERE CAN AND PERHAPS SHOULD BE AN INTONATION BREAK
        BETWEEN THE TWO BIGNUMS'

    EXAMPLES
        FIFTY TWO THOUSAND FOUR HUNDRED (OK)
        THREE HUNDRED TWO THOUSAND (OUT) --
            BECAUSE THOUSAND IS TO BE MULTIPLIED BY
            THREE HUNDRED TWO, NOT ADDED TO THREE HUNDRED TWO
```

```
            SEE RULEDEF NUM 16;

 END;


RULE.DEF NUM16    BIGMULT = BIGMULT BIGCAT;

    ATTRIBUTES
        NUM = SMULT(NUM(BIGMULT),NUM(BIGCAT));

    FACTORS
        NUM = [X=NUM(BIGMULT),Y=NUM(BIGCAT),
            IF NUMBERP(X) AND NUMBERP(Y) THEN
            IF X LS Y THEN OK ELSE OUT ELSE OK];
        %'AN INFLECTION BREAK HERE IS VERY UNLIKELY'

    EXAMPLES
        THREE HUNDRED TWO THOUSAND (OK)
        FIFTY TWO THOUSAND THREE HUNDRED (OUT) --
            BECAUSE HUNDRED IS NOT TO BE MULTIPLIED BY
            FIFTY TWO THOUSAND THREE, INSTEAD, THREE HUNDRED
            IS TO BE ADDED TO FIFTY TWO THOUSAND
            SEE RULEDEF NUM15;

 END;


EOF
```

APPENDIX B REPORTS AND PUBLICATIONS

Becker, Richard, and Poza, Fausto. Acoustic Processing in the SRI Speech Understanding System. IEEE Transactions on Acoustics, Speech and Signal Processing, 1975 [in press].

Deutsch, Barbara G. The Structure of Task-Oriented Dialogs. Contributed Papers, IEEE Symposium on Speech Recognition, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 15-19 April 1974. IEEE, New York, 1974, 250-254.

Hendrix, Gary G. Expanding the Utility of Semantic Networks Through Partitioning. Advance Papers, International Joint Conference on Artificial Intelligence, Tbilisi, Georgian SSR, 3-8 September 1975.

Paxton, William H. A Best-First Parser. Contributed Papers, IEEE Symposium on Speech Recognition, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 15-19 April 1974. [IEEE, New York, 218-225. IEEE Transactions on Acoustics, Speech and Signal Processing, in press.]

Paxton, William H., and Robinson, Ann E. A Parser for a Speech Understanding System. Advance Papers, International Joint Conference on Artificial Intelligence, Stanford, California, 20-23 August 1973. Stanford Research Institute, Menlo Park, California, 1973, 216-222.

Robinson, Jane J. Performance Grammars. Invited Papers, IEEE Symposium on Speech Recognition, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 15-19 April 1974. To be published by Academic Press, New York, 1975.

Walker, Donald E. Speech Understanding Research. Annual Report, Project 1526, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California, February 1973.

Walker, Donald E. Speech Understanding Through Syntactic and Semantic Analysis. Advance Papers, International Joint Conference on Artificial Intelligence, Stanford, California, 20-23 August 1973. Stanford Research Institute, Menlo Park, California, 1973, 208-215.

Walker, Donald E. Speech Understanding, Computational Linguistics, and Artificial Intelligence. In Computational and Mathematical Linguistics, Proceedings of the International Conference on Computational Linguistics, Volume I. Edited by Antonio Zampolli. Casa Editrice Leo S. Olschki, Firenze, 1973.

Walker, Donald E. Speech Understanding Research. Annual Report, Project 1526, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California, May 1974.

Walker, Donald E. The SRI Speech Understanding System. Contributed Papers, IEEE Symposium on Speech Recognition, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 15-19 April 1974. [IEEE, New York, 32-37. IEEE Transactions on Acoustics, Speech and Signal Processing, in press.]

Walker, Donald E., et al. Speech Understanding Research. Annual Report, Project 3804, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California, June 1975.